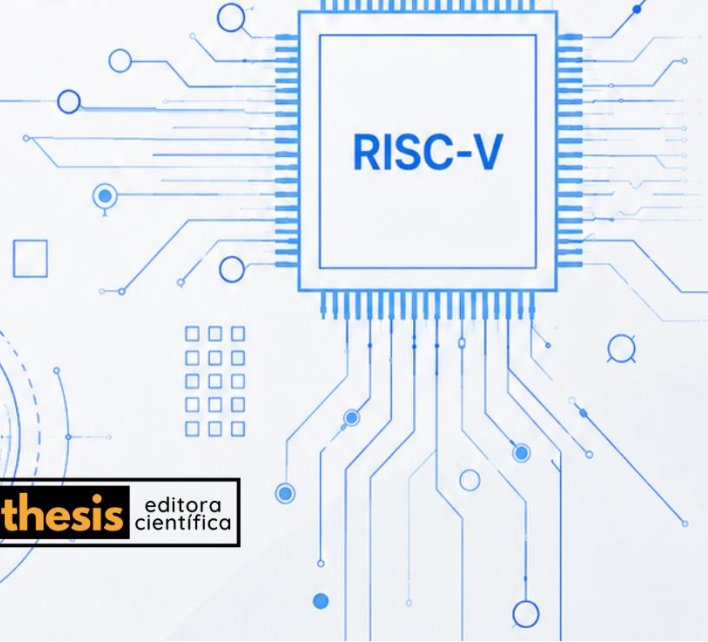
RISC-V:

Viabilidade e adoção em hardware embarcado

Alberto José Oliveira Pereira



thesis editora científica

Editor Chefe: Felipe Cardoso Rodrigues Vieira.
Diagramação e Projeto Gráfico: Thesis Ed. Cien.
Design da Capa: Alberto José Oliveira Pereira
Revisão: Alberto José Oliveira Pereira

ISBN: 978-65-83199-68-3

Thesis Editora Científica
Teresina – PI – Brasil
contato@thesiseditora.com.br
www.thesiseditora.com.br

RISC-V

Viabilidade e adoção em hardware embarcado

Fundamentos, ecossistema, critérios de engenharia e perspectivas

Alberto José Oliveira Pereira

Belo Horizonte · 2026

CONSELHO EDITORIAL

Editor Chefe: Felipe Cardoso Rodrigues Vieira.
Diagramação e Projeto Gráfico: Thesis Ed. Cien.
Design da Capa: Alberto José Oliveira Pereira
Revisão: Alberto José Oliveira Pereira

Catalogação na publicação
Elaborada por Bibliotecária Janaina Ramos – CRB-8/9166

P436r

Pereira, Alberto José Oliveira

RISC-V: viabilidade e adoção em hardware embarcado / Alberto José Oliveira Pereira. – Teresina-PI: Thesis Editora Científica, 2026.

Livro em PDF
ISBN 978-65-83199-68-3

1. RISC-V (Arquitetura de computadores). 2. Hardware. 3. Engenharia de computação. I. Pereira, Alberto José Oliveira. II. Título.

CDD 004.22

Índice para catálogo sistemático

I. RISC-V (Arquitetura de computadores)

Thesis Editora Científica
Teresina – PI – Brasil
contato@thesiseditora.com.br
www.thesiseditora.com.br

DEDICATÓRIA

Dedico esta obra, com profundo amor e gratidão, ao meu pai, Sebastião (in memoriam), cuja memória permanece viva em meu coração; à minha mãe, Orli, por sua dedicação, amor e cuidado incansável com os filhos; e às minhas irmãs, Ana e Alessandra, pela presença, carinho e companheirismo ao longo da vida.

À minha família, meu eterno agradecimento por ser fonte de força, inspiração e incentivo em cada etapa desta caminhada.

SOBRE O AUTOR

Alberto José Oliveira Pereira é formado em Biblioteconomia (UFMG), Arquivologia (UFMG) e Engenharia da Computação (FUMEC). Sua atuação integra as áreas de engenharia, sistemas computacionais e interação humano-computador, aplicadas à gestão arquivística e à preservação da informação em ambientes digitais. No âmbito acadêmico e profissional, dedica-se ao estudo de arquitetura de computadores, sistemas embarcados, desenvolvimento de software, preservação digital e tecnologias de código aberto.

APRESENTAÇÃO

Esse livro é fruto da ampliação e da revisão substantiva do estudo acadêmico “RISC-V: estudo da viabilidade da adoção em hardware embarcado” — apresentado ao curso de Engenharia de Computação da Universidade FUMEC em 2024 —, esta obra preserva a pergunta que lhe deu origem: em que condições a arquitetura ***RISC-V se configura como alternativa viável para sistemas embarcados?*** Seu escopo, porém, não se limita àquela investigação original: a edição amplia-se para funcionar como referência introdutória e técnica a estudantes, engenheiros e profissionais que atuam no desenvolvimento de hardware e firmware.

A transformação de um trabalho de conclusão de curso em livro, contudo, não se restringe a uma atualização de projeto gráfico. O conteúdo foi reestruturado para uma leitura autônoma, com capítulos inéditos que abrangem desde os fundamentos da arquitetura de conjunto de instruções e a modularidade do RISC-V até o ecossistema de software — compiladores, sistemas operacionais e ferramentas de depuração —, sem deixar de lado aspectos estratégicos como critérios de seleção, migração e perspectivas de evolução. A metodologia e os resultados do estudo original permanecem como núcleo analítico da obra, enquanto os novos capítulos oferecem o suporte técnico e operacional necessário à sua contextualização.

Diante da rápida evolução do RISC-V, a edição também se preocupa em demarcar o que é duradouro — a ISA, a microarquitetura, as extensões e os modelos de privilégio — daquilo que depende do

estágio atual do ecossistema. No plano da prática profissional, essa distinção é crucial: a decisão de adoção exige a ponderação sobre a implementação concreta do núcleo, o processo de fabricação, a maturidade das ferramentas e do software disponível, os requisitos de segurança funcional, o suporte do fornecedor e, por fim, o custo total de propriedade ao longo do ciclo de vida.

PREFÁCIO

Arquiteturas de processadores são frequentemente tratadas como se correspondessem a produtos únicos e diretamente comparáveis. Essa simplificação, entretanto, é particularmente inadequada no caso do RISC-V. Mais do que um processador ou uma família específica de processadores, o RISC-V é uma arquitetura aberta e modular de conjunto de instruções (ISA — Instruction Set Architecture). Diferentes processadores compatíveis com a mesma ISA podem apresentar diferenças significativas em aspectos como pipeline, hierarquia de memória e caches, unidades funcionais, frequência de operação, consumo de energia, mecanismos de segurança, recursos de depuração e integração de periféricos. Por essa razão, comparações tecnicamente consistentes devem distinguir claramente a ISA da implementação microarquitetural.

Essa distinção é especialmente relevante no contexto dos sistemas embarcados. Nesse domínio, o engenheiro não seleciona simplesmente um conjunto de instruções ou uma família de processadores, mas uma plataforma tecnológica completa, que deverá atender simultaneamente a requisitos de desempenho, tempo real, consumo de energia, capacidade de memória, confiabilidade, segurança, custo, certificação, disponibilidade de componentes, cadeia de suprimentos e manutenção ao longo do ciclo de vida do produto.

Nesse contexto, o objetivo desta obra é estabelecer uma estrutura de análise que permita compreender de forma crítica em quais situações as características de abertura, modularidade e extensibilidade do

RISC-V podem representar vantagens concretas e, por outro lado, em quais cenários a maturidade, a disponibilidade de ferramentas, o ecossistema e a experiência acumulada por outras arquiteturas ainda podem constituir fatores determinantes.

A proposta não é declarar uma arquitetura universalmente superior às demais. Em engenharia, escolhas tecnológicas devem ser fundamentadas nos requisitos e nas restrições específicas de cada projeto. Em aplicações de baixo custo, alto grau de customização ou que demandem maior controle sobre a arquitetura do processador, o RISC-V pode reduzir barreiras de entrada e ampliar as possibilidades de inovação e diferenciação tecnológica. Em aplicações críticas, entretanto, a abertura da ISA, por si só, não constitui garantia de adequação. Requisitos de segurança funcional, segurança cibernética, confiabilidade, suporte técnico, validação, rastreabilidade e certificação continuam exigindo evidências concretas.

Assim, a escolha de uma arquitetura de processador deve ser compreendida como uma decisão de engenharia, e não como uma disputa entre tecnologias. O RISC-V oferece novas possibilidades, mas seus benefícios e limitações precisam ser avaliados à luz do contexto de aplicação. A arquitetura mais adequada será, portanto, aquela que melhor atender aos requisitos técnicos, econômicos e operacionais estabelecidos para o sistema.

RESUMO

Este livro analisa a viabilidade da adoção da arquitetura RISC-V em hardware embarcado, considerando desempenho, consumo de energia, confiabilidade, segurança, custo, flexibilidade de projeto e maturidade do ecossistema. A obra parte de estudo comparativo realizado entre RISC-V, ARM e x86 e amplia a discussão para incluir a natureza modular da ISA, extensões padronizadas, perfis, ferramentas de desenvolvimento, sistemas operacionais, depuração, segurança e estratégias de migração. O argumento central é que a viabilidade do RISC-V não pode ser determinada apenas pela abertura da ISA ou por benchmarks isolados: ela depende da implementação selecionada, do software disponível, dos requisitos de ciclo de vida e da capacidade da organização de validar a plataforma. O RISC-V apresenta vantagens relevantes em personalização, independência tecnológica e possibilidade de otimização por domínio, enquanto plataformas consolidadas continuam oferecendo benefícios em maturidade, suporte e certificações. A decisão de adoção deve, portanto, ser orientada por uma matriz de requisitos técnicos, econômicos e operacionais.

Palavras-chave: RISC-V; arquitetura de computadores; sistemas embarcados; ISA; microcontroladores; eficiência energética; hardware aberto; engenharia de computação.

SUMÁRIO

DEDICATÓRIA 6
SOBRE O AUTOR 7
APRESENTAÇÃO 8
PREFÁCIO 10
RESUMO 12
SUMÁRIO 13
1 INTRODUÇÃO E PROBLEMA DE ENGENHARIA 15
2 SISTEMAS EMBARCADOS E CRITÉRIOS DE PROJETO 18
3 FUNDAMENTOS DE ARQUITETURAS DE CONJUNTO DE INSTRUÇÕES 21
4 ARQUITETURA RISC-V: BASE, EXTENSÕES E MODULARIDADE 23
5 ECOSSISTEMA DE DESENVOLVIMENTO: COMPILADORES, DEPURAÇÃO E SISTEMAS OPERACIONAIS 26
6 METODOLOGIA DO ESTUDO COMPARATIVO 29
7 RESULTADOS DO ESTUDO ORIGINAL E RELEITURA TÉCNICA 30
7.1 Estrutura de evidências e níveis de comparação 30
7.2 Desempenho: do CPI ao tempo de resposta da aplicação 32
7.3 Energia: separar potência instantânea de energia por tarefa 33
7.4 Confiabilidade, segurança e disponibilidade 34
7.5 Custo: licença da ISA não é custo total do produto 35
7.6 Ecossistema como resultado de engenharia 36
7.7 Síntese dos resultados 37
8 DISCUSSÃO: DA COMPARAÇÃO DE ARQUITETURAS À DECISÃO DE ENGENHARIA 38
8.1 ISA, microarquitetura e produto: três conceitos que não devem ser confundidos 38
8.2 Modularidade da RISC-V como ferramenta de projeto 39

8.3 O estado das especificações e a importância de versões ratificadas ... 39
8.4 Toolchain, ABI e reprodutibilidade ... 40
8.5 Sistema operacional e RTOS ... 41
8.6 Segurança por projeto: mecanismos, configuração e ciclo de atualização ... 42
8.7 Aplicações de baixo consumo e IoT ... 42
8.8 Edge AI e extensões especializadas ... 43
8.9 Cadeia de suprimentos, longevidade e risco de fornecedor ... 44
8.10 Estratégia experimental recomendada ... 44
8.11 Critérios de decisão por classe de produto ... 45
8.12 Síntese da discussão ... 46
9 SEGURANÇA, PRIVILÉGIO E ISOLAMENTO EM PLATAFORMAS RISC-V ... 47
10 RISC-V EM APLICAÇÕES EMBARCADAS E DE BORDA ... 49
11 ESTRATÉGIA DE ADOÇÃO E MIGRAÇÃO PARA RISC-V ... 51
12 MATRIZ DE DECISÃO PARA SELEÇÃO DE ARQUITETURA ... 53
13 LIMITAÇÕES, RISCOS E PERSPECTIVAS DE EVOLUÇÃO ... 55
14 BENCHMARKS E MEDIÇÕES PARA SISTEMAS EMBARCADOS 57
15 PORTABILIDADE DE FIRMWARE E QUALIDADE DE SOFTWARE 60
16 SELEÇÃO DE NÚCLEO, SOC E PLATAFORMA ... 63
17 PADRÃO ABERTO, HARDWARE ABERTO E MODELOS DE LICENCIAMENTO ... 66
18 ROADMAP DE ADOÇÃO ORGANIZACIONAL ... 69
19 CONCLUSÕES ... 72
APÊNDICE A – ROTEIRO DE AVALIAÇÃO TÉCNICA ... 74
APÊNDICE B – GLOSSÁRIO ESSENCIAL ... 76
REFERÊNCIAS ... 79
APÊNDICE C – QUADROS E TABELAS DO ESTUDO ORIGINAL ... 86

1 INTRODUÇÃO E PROBLEMA DE ENGENHARIA

Sistemas embarcados estão presentes em produtos de consumo, equipamentos industriais, veículos, dispositivos médicos, infraestrutura de comunicação e sistemas de Internet das Coisas. Embora sejam computadores, diferem de plataformas de uso geral por serem projetados para executar funções específicas sob restrições de energia, custo, espaço físico, temperatura, confiabilidade e tempo de resposta. Essas restrições fazem com que a escolha do processador seja uma decisão de arquitetura do sistema, e não apenas uma decisão de desempenho.

O estudo que deu origem a esta obra formulou uma pergunta objetiva: a arquitetura RISC-V é uma solução viável para hardware embarcado em comparação com alternativas consolidadas? Para responder, foram considerados quatro eixos: desempenho, consumo de energia, confiabilidade e custo. Nesta edição ampliada, a pergunta é mantida, mas ganha dimensões adicionais que se tornaram indispensáveis para a prática de engenharia: maturidade do ecossistema, portabilidade de software, depuração, segurança, suporte de longo prazo e possibilidade de certificação.

RISC-V surgiu no ambiente acadêmico de Berkeley e tornou-se uma arquitetura de conjunto de instruções aberta, padronizada e extensível. A abertura da especificação cria condições para que universidades, empresas e fabricantes desenvolvam implementações próprias sem depender de uma ISA proprietária. Isso não significa que todo

processador RISC-V seja “open source”, nem que todo produto baseado em RISC-V seja isento de custos. A ISA pode ser aberta enquanto núcleos, SoCs, ferramentas ou serviços associados permanecem proprietários. Essa distinção é fundamental para uma análise econômica correta.

Outro ponto central é evitar a comparação direta entre “RISC-V” e um processador específico como se fossem entidades equivalentes. RISC-V define a interface arquitetural visível ao software. O desempenho e o consumo reais são determinados por decisões de microarquitetura e implementação física: profundidade do pipeline, predição de desvios, tamanho de cache, largura de execução, tecnologia de fabricação, frequência, tensão, aceleradores e subsistemas de memória. Assim, a escolha deve ser feita entre plataformas e implementações concretas, usando a ISA como um dos critérios.

O objetivo geral desta edição é oferecer uma visão técnica que permita avaliar a adoção do RISC-V em sistemas embarcados com base em requisitos verificáveis. Os objetivos específicos são: explicar os fundamentos da arquitetura; apresentar sua organização modular; relacionar as extensões aos tipos de aplicação; examinar o ecossistema de compiladores e sistemas operacionais; discutir segurança e depuração; reapresentar o estudo comparativo original; e propor um roteiro de decisão aplicável a projetos reais.

A relevância do tema decorre da necessidade crescente de autonomia tecnológica e de especialização de hardware. Processamento de borda,

aprendizado de máquina embarcado, sensores inteligentes, automação e dispositivos conectados exigem soluções capazes de equilibrar desempenho e energia. Uma ISA modular favorece implementações adaptadas ao domínio, mas essa liberdade traz responsabilidade adicional de validação. O engenheiro deve compreender tanto as oportunidades quanto os riscos.

2 SISTEMAS EMBARCADOS E CRITÉRIOS DE PROJETO

Um sistema embarcado é uma plataforma computacional integrada a um produto ou processo maior e orientada a um conjunto definido de funções. Essa definição abrange desde microcontroladores de poucos recursos até sistemas complexos com múltiplos núcleos, memória virtual, aceleradores e sistemas operacionais completos. O elemento comum é a relação direta entre a computação e a função do equipamento.

Em projetos embarcados, desempenho raramente pode ser analisado isoladamente. Uma solução mais rápida pode ser inadequada se consumir energia excessiva, exigir refrigeração, aumentar o custo da placa ou comprometer autonomia de bateria. Da mesma forma, a solução de menor consumo pode falhar se não satisfizer prazos de tempo real. O processo de seleção deve começar pela especificação de requisitos e pela definição de métricas mensuráveis.

Os requisitos podem ser organizados em seis grupos. O primeiro é computacional: frequência de amostragem, carga de processamento, latência, throughput, interrupções e paralelismo. O segundo é energético: potência média, picos de corrente, modos de sono e tempo de recuperação. O terceiro envolve memória e armazenamento. O quarto trata de conectividade e periféricos. O quinto cobre confiabilidade, segurança e disponibilidade. O sexto abrange custo, cadeia de suprimentos, suporte e longevidade do produto.

Aplicações de tempo real exigem atenção especial à previsibilidade. Um processador pode apresentar alta pontuação em benchmark e ainda assim produzir latências indesejáveis em interrupções ou acessos à memória. Para controladores industriais, automotivos e sistemas de aquisição, o comportamento no pior caso pode ser mais importante que o desempenho médio. Por isso, medições devem incluir jitter, latência de interrupção e interferência de cache e barramento.

Em dispositivos alimentados por bateria, o consumo total depende não apenas da eficiência por instrução, mas do perfil de atividade. Um núcleo que conclui uma tarefa rapidamente e entra em sono profundo pode consumir menos energia que outro aparentemente mais eficiente durante execução contínua. A análise deve considerar duty cycle, periféricos, reguladores, memória, rádio e tempo em modos de baixa potência.

A confiabilidade também deve ser definida em função do ambiente e da criticidade. Um sensor doméstico e um controlador médico não possuem o mesmo nível de exigência. Temperatura, vibração, radiação, interferência eletromagnética, retenção de memória, detecção de falhas e atualização segura de firmware podem transformar a escolha do processador. Em muitos casos, a disponibilidade de documentação, mecanismos de diagnóstico e histórico de suporte pesa mais que diferenças pequenas de desempenho.

Finalmente, custo deve ser entendido como custo total de propriedade. Preço unitário do componente é apenas uma parcela. Entram na conta licenças, ferramentas, treinamento, redesign de placa, manutenção de software, testes, certificações, estoque, suporte e risco de descontinuação. A promessa de redução de royalties do RISC-V é relevante, mas deve ser inserida nessa visão mais ampla.

3 FUNDAMENTOS DE ARQUITETURAS DE CONJUNTO DE INSTRUÇÕES

A arquitetura de conjunto de instruções, ou ISA, define o contrato entre software e processador. Ela especifica instruções, registradores, tipos de dados, comportamento de exceções e outros elementos observáveis pelo programa. A microarquitetura, por outro lado, descreve como uma implementação realiza esse contrato. Essa separação permite que processadores diferentes executem o mesmo software binário mesmo usando pipelines, caches e unidades internas distintas.

A distinção histórica entre RISC e CISC ajuda a compreender decisões de projeto, mas deve ser usada com cuidado. Processadores modernos combinam técnicas que tornam a fronteira menos rígida. O valor do conceito RISC está na busca por uma interface regular, operações simples e mecanismos que favorecem implementação eficiente. Já arquiteturas tradicionalmente classificadas como CISC preservam conjuntos de instruções ricos e compatibilidade histórica, frequentemente traduzindo internamente instruções complexas em operações menores.

Em um sistema embarcado, a ISA influencia o compilador, a densidade de código, os mecanismos de interrupção, o acesso a recursos privilegiados e o suporte a extensões especializadas. Entretanto, a ISA não determina sozinha a eficiência energética ou a segurança de um chip. Recursos como ECC, TrustZone, enclaves, PMP, aceleradores criptográficos e boot seguro podem estar

associados a uma plataforma específica e precisam ser avaliados individualmente.

O conceito de ABI, interface binária de aplicação, complementa a ISA. A ABI define convenções de chamada, uso de registradores, representação de tipos, alinhamento e formato necessário para que compiladores, bibliotecas e sistemas operacionais interoperem. Para uma organização que pretende migrar software, a estabilidade da ABI e o suporte das ferramentas são tão importantes quanto a existência das instruções.

Outro elemento essencial é a densidade de código. Sistemas embarcados frequentemente possuem memória flash limitada. Conjuntos de instruções comprimidas podem reduzir tamanho do binário e, indiretamente, tráfego de memória. No RISC-V, extensões padronizadas podem adicionar instruções comprimidas e outras capacidades sem obrigar todas as implementações a incorporarem o mesmo conjunto completo de funcionalidades.

A modularidade introduz uma vantagem e uma dificuldade. A vantagem é permitir que o processador seja configurado conforme o domínio. A dificuldade é gerenciar compatibilidade: software compilado para determinada combinação de extensões exige que o hardware ofereça essas extensões ou que exista uma estratégia de fallback. Por isso, perfis e convenções de plataforma tornam-se importantes para reduzir fragmentação.

4 ARQUITETURA RISC-V: BASE, EXTENSÕES E MODULARIDADE

RISC-V organiza a arquitetura em conjuntos de instruções base e extensões. Essa abordagem permite iniciar com um núcleo mínimo e acrescentar recursos conforme a aplicação. Em sistemas embarcados, a possibilidade de selecionar funcionalidades é atraente porque áreas de silício e energia podem ser direcionadas para recursos efetivamente utilizados. Ao mesmo tempo, a seleção precisa considerar o software que será executado e as expectativas do ecossistema.

As bases RV32 e RV64 representam espaços de registradores e endereçamento adequados a diferentes classes de sistema. Microcontroladores de baixa complexidade podem utilizar implementações de 32 bits, enquanto plataformas embarcadas avançadas e sistemas capazes de executar ambientes de aplicação completos podem adotar 64 bits. A escolha não deve ser orientada apenas por “mais bits”, mas por necessidade de memória, capacidade computacional, software e custo.

Extensões adicionam recursos como multiplicação e divisão, ponto flutuante, operações atômicas, instruções comprimidas, vetores e operações específicas. Para firmware de controle, um conjunto enxuto pode ser suficiente. Para processamento de sinais, criptografia ou aprendizado de máquina, extensões especializadas e aceleradores podem reduzir tempo de execução e energia. O ganho, porém, precisa ser medido na implementação real e no código-alvo.

A especificação privilegiada define mecanismos utilizados por sistemas operacionais, hipervisores e ambientes com diferentes níveis de privilégio. Em microcontroladores simples, nem todos esses recursos são necessários. Em plataformas capazes de executar Linux ou de isolar múltiplos componentes de software, mecanismos privilegiados e de proteção de memória assumem importância maior.

Perfis padronizados procuram estabelecer conjuntos previsíveis de recursos para classes de plataformas. Isso é especialmente relevante quando o objetivo é distribuir software binário para múltiplos fabricantes. Em sistemas profundamente embarcados, o grau de customização pode continuar alto, mas projetos que dependem de sistemas operacionais e pacotes comuns se beneficiam de bases mais uniformes.

A abertura do padrão também favorece extensões personalizadas. Um fabricante pode adicionar instruções específicas para seu domínio, desde que administre adequadamente a compatibilidade. A decisão deve considerar benefício mensurável, custo de manutenção do compilador e depurador, risco de dependência de uma extensão proprietária e impacto na portabilidade. Uma extensão customizada é tecnicamente poderosa, mas cria uma nova interface que a organização passará a manter.

Para fins de engenharia, a análise de uma plataforma RISC-V deve registrar explicitamente: base ISA, extensões obrigatórias, extensões opcionais, nível de privilégio, mecanismos de proteção, recursos de

depuração e interface de interrupções. Esse inventário evita pressupostos e facilita a comparação entre componentes.

5 ECOSSISTEMA DE DESENVOLVIMENTO: COMPILADORES, DEPURAÇÃO E SISTEMAS OPERACIONAIS

A viabilidade de uma arquitetura para sistemas embarcados depende diretamente do ecossistema de software. Um processador sem compilador estável, bibliotecas, depurador e suporte de sistema operacional pode elevar custos de engenharia mesmo quando o hardware é atraente. RISC-V ganhou suporte em toolchains amplamente utilizados, o que reduz a barreira de entrada, mas a qualidade final depende da combinação entre extensão, fornecedor e versão das ferramentas.

GCC e LLVM/Clang são pilares do desenvolvimento em C e C++ para RISC-V. O compilador precisa conhecer a combinação exata de ISA e ABI selecionada para gerar código adequado. Em projetos profissionais, parâmetros de compilação devem ser versionados no sistema de build, evitando que desenvolvedores gerem binários incompatíveis com a plataforma. O mesmo cuidado vale para bibliotecas de tempo de execução e para a implementação da linguagem C utilizada.

Depuração é outro componente crítico. Desenvolvimento embarcado requer breakpoints, acesso a registradores, inspeção de memória, controle de execução e, em muitos casos, trace. A existência de uma especificação de debug padronizada é importante, mas o fluxo real depende da implementação do SoC, do probe utilizado, de OpenOCD ou ferramentas do fabricante e do ambiente de desenvolvimento.

Antes de adotar uma plataforma, deve-se executar um teste completo de flash, reset, breakpoints, watchpoints e diagnóstico de exceções.

Sistemas operacionais em tempo real reduzem o esforço de criação de drivers, escalonamento, sincronização e rede. O ecossistema RISC-V possui suporte em projetos de RTOS e também em sistemas operacionais de uso geral, mas é necessário verificar o suporte para a placa e os periféricos concretos. Dizer que um sistema operacional "suporta RISC-V" não garante que um determinado SoC possua drivers maduros para todos os blocos de hardware.

Para projetos bare metal, a cadeia de inicialização merece atenção. O startup deve configurar pilha, memória, seções de dados, vetor ou mecanismo de interrupções e, quando aplicável, clock e proteção de memória. Em arquiteturas novas para a equipe, erros nessa camada podem ser confundidos com defeitos de hardware ou compilador. Um pacote de suporte do fabricante bem mantido reduz o risco.

A disponibilidade de emuladores e plataformas virtuais também é relevante. QEMU e simuladores de ISA possibilitam executar software antes da chegada do hardware, automatizar testes e estudar comportamento arquitetural. Entretanto, simulação funcional não substitui validação temporal, energética ou elétrica na plataforma final.

Uma estratégia madura de adoção deve incluir integração contínua para compilar firmware em versões controladas da toolchain, executar testes unitários no host, testes funcionais em simuladores quando

possível e testes hardware-in-the-loop em placas reais. Dessa forma, a escolha de uma nova arquitetura passa a ser administrada como uma mudança de plataforma, e não como uma simples troca de compilador.

6 METODOLOGIA DO ESTUDO COMPARATIVO

A pesquisa original foi conduzida de forma exploratória e comparativa, combinando análise qualitativa e quantitativa para avaliar a viabilidade do RISC-V em hardware embarcado. Foram consultadas bases acadêmicas e documentos técnicos, com foco em desempenho, consumo de energia, confiabilidade e custo. Nesta edição, a metodologia é reapresentada como estudo de base e deve ser interpretada no contexto temporal das fontes utilizadas em 2024.

Uma limitação importante de estudos comparativos por arquitetura é a heterogeneidade das implementações. Por isso, resultados numéricos associados a processadores específicos não devem ser generalizados para toda a ISA. Ao reutilizar tabelas e achados do estudo original, esta edição mantém a finalidade comparativa e recomenda que decisões de projeto sejam confirmadas por benchmarks reproduzidos no hardware candidato.

As métricas fundamentais adotadas são: desempenho; consumo de energia; confiabilidade; custo. Para uma aplicação profissional, recomenda-se complementar essas métricas com latência de interrupção, footprint de memória, tempo de boot, cobertura de drivers, segurança, suporte de longo prazo, disponibilidade de componentes e esforço de migração.

7 RESULTADOS DO ESTUDO ORIGINAL E RELEITURA TÉCNICA

O estudo original organizou a comparação entre RISC-V, ARM e, em pontos específicos, x86 a partir de quatro dimensões centrais: desempenho, consumo de energia, confiabilidade e custo. A edição ampliada preserva esse núcleo, mas acrescenta uma releitura metodológica necessária para evitar uma simplificação frequente em comparações de arquiteturas: a ISA não é, por si só, equivalente ao processador físico. RISC-V define uma interface arquitetural e um conjunto de instruções; desempenho, potência, área, segurança implementada e confiabilidade emergem da microarquitetura, do processo de fabricação, da hierarquia de memória, dos periféricos, das extensões adotadas, do software e do contexto de uso.

Essa distinção modifica a maneira de interpretar os resultados. Uma comparação válida não deve concluir que "RISC-V consome menos" ou que "ARM é mais segura" como propriedades universais. O que pode ser afirmado é que implementações concretas, sob determinadas cargas de trabalho e condições, apresentam resultados mensuráveis. A função do engenheiro é identificar quais características decorrem da ISA e quais pertencem ao produto ou à plataforma selecionada.

7.1 Estrutura de evidências e níveis de comparação

Para tornar a análise reproduzível, os resultados podem ser classificados em quatro níveis. O primeiro é o nível da ISA, no qual se observam codificação de instruções, registradores, modos de privilégio e extensões padronizadas. O segundo é o nível da

microarquitetura, que inclui pipeline, predição de desvios, caches, execução em ordem ou fora de ordem e unidades especializadas. O terceiro é o nível do SoC, no qual entram controladores de memória, aceleradores, interfaces, timers, controladores de interrupção e mecanismos de segurança. O quarto é o nível da plataforma, que inclui placa, alimentação, memória externa, sistema operacional, compilador, bibliotecas e firmware.

Nível	**Objeto avaliado**	**Exemplos de métricas**	**Risco de interpretação**
ISA	Conjunto de instruções e extensões	densidade de código, recursos arquiteturais, ABI	atribuir desempenho físico diretamente à ISA
Microarquitetura	Implementação do núcleo	IPC/CPI, frequência, latência, caches	comparar núcleos de classes diferentes
SoC	Sistema em chip	potência, periféricos, boot, segurança	ignorar aceleradores e controladores integrados
Plataforma	Hardware + software	energia da aplicação, tempo total, memória, estabilidade	medir placas diferentes como se fossem apenas CPUs

Essa decomposição é especialmente importante no caso de RISC-V devido à liberdade de implementação. Dois processadores compatíveis com RV32 podem ter perfis de desempenho e consumo radicalmente distintos. A modularidade permite selecionar extensões e construir núcleos minimalistas ou sofisticados, o que é uma vantagem de projeto, mas exige maior cuidado ao generalizar resultados.

7.2 Desempenho: do CPI ao tempo de resposta da aplicação

O trabalho original utilizou indicadores de desempenho e referências a benchmarks para discutir competitividade. Na edição ampliada, o critério principal passa a ser o tempo necessário para concluir uma carga de trabalho representativa. CPI isolado é útil para compreender o comportamento do núcleo, mas não captura frequência, stalls de memória, efeitos de cache, periféricos, DMA, compilação, interrupções ou espera por E/S. Em sistemas embarcados, o tempo de resposta fim a fim costuma ser mais relevante que um índice abstrato do processador.

Uma avaliação prática deve registrar a versão do compilador, opções de otimização, ABI, string de arquitetura utilizada na compilação, frequência real do clock e política de energia. Também deve distinguir testes sintéticos de testes de aplicação. CoreMark e Dhrystone podem fornecer indicadores de referência, porém não substituem cargas reais, como filtragem de sensores, criptografia, inferência TinyML, compressão, controle em tempo real ou processamento de protocolos.

Indicador	**O que mede**	**Uso recomendado**	**Limitação principal**
Tempo de execução	latência total de uma tarefa	comparar aplicações reais	depende de toda a plataforma
Throughput	trabalho por unidade de tempo	streaming, rede, processamento contínuo	pode ocultar piores latências
CPI/IPC	eficiência interna de execução	análise microarquitetural	não representa sozinho o tempo final
Tamanho do binário	densidade de código	flash limitada e OTA	não mede velocidade
Latência de interrupção	resposta a eventos	controle e tempo real	sensível a RTOS e configuração

Para aplicações de tempo real, recomenda-se observar não apenas médias, mas também piores casos e dispersão. Uma plataforma que apresenta alta média de desempenho pode ser inadequada se produzir caudas de latência incompatíveis com deadlines. O mesmo raciocínio vale para interrupções, acesso a flash, caches e memória externa.

7.3 Energia: separar potência instantânea de energia por tarefa

A comparação energética deve distinguir potência, energia e eficiência. Potência é a taxa instantânea de consumo; energia representa o consumo acumulado ao longo do tempo. Um processador pode consumir mais potência durante alguns milissegundos e, ainda assim, executar a tarefa tão rapidamente que a energia total seja inferior. Por isso, a métrica energia por operação ou energia por tarefa é particularmente útil em dispositivos alimentados por bateria.

Outra variável relevante é o duty cycle. Sensores IoT podem permanecer a maior parte do tempo em suspensão e despertar apenas para medir, processar e transmitir dados. Nesse cenário, corrente de sleep, tempo de wake-up, retenção de memória e eficiência dos periféricos podem dominar o orçamento energético. A análise deve, portanto, incluir períodos ativos e inativos e não apenas consumo do núcleo em carga.

Fase	**Métrica**	**Instrumentação sugerida**	**Observação**

Fase	Métrica	Instrumentação sugerida	Observação
Sleep	corrente média de repouso	medidor de corrente/energia	verificar periféricos e reguladores
Wake-up	tempo e energia de retomada	osciloscópio ou power profiler	inclui inicializações
Processamento	energia por tarefa	integração de potência no tempo	comparar mesma carga
Comunicação	energia por pacote/transação	perfilador + logs	rádio pode dominar o consumo
Ciclo completo	mWh ou J por ciclo funcional	medição de longa duração	melhor indicador de autonomia

A modularidade RISC-V pode favorecer implementações sob medida, porém a vantagem precisa ser demonstrada na plataforma escolhida. Extensões especializadas podem reduzir o número de instruções ou acelerar operações, mas também podem aumentar área, complexidade e consumo estático. A decisão correta depende do perfil de carga.

7.4 Confiabilidade, segurança e disponibilidade

No manuscrito original, confiabilidade foi discutida por meio de recursos como ECC, tolerância a falhas, secure boot e suporte a ambientes críticos. A edição ampliada faz uma distinção adicional: confiabilidade é uma propriedade do sistema e não pode ser inferida somente pelo nome da ISA. ECC, redundância, watchdogs, lockstep, memória protegida, diagnósticos e certificações dependem da implementação e do produto.

Segurança também exige análise por camadas. A arquitetura privilegiada RISC-V padroniza mecanismos de execução privilegiada e controle do sistema, enquanto plataformas específicas podem acrescentar PMP, raízes de confiança, secure boot, armazenamento de chaves, aceleradores criptográficos e isolamento de periféricos. A existência de um mecanismo na especificação não garante que um determinado microcontrolador o implemente, configure ou certifique de forma adequada.

Para projetos críticos, o processo de seleção deve partir dos requisitos: nível de integridade, cobertura de diagnóstico, comportamento em falhas, suporte do fabricante, histórico de erratas, disponibilidade de documentação e estratégia de atualização. A abertura da ISA é vantajosa para inspeção e customização, mas não elimina a necessidade de validação de hardware e software.

7.5 Custo: licença da ISA não é custo total do produto

Uma das motivações mais conhecidas para a adoção de RISC-V é seu modelo de padrão aberto. Entretanto, ausência de royalties associados ao uso da ISA não significa custo zero. O custo total de propriedade inclui engenharia, verificação, IPs complementares, ferramentas, integração, suporte, fabricação, certificação, manutenção e qualificação da cadeia de suprimentos. Para quem compra um microcontrolador pronto, o preço unitário e o custo de migração podem ser mais relevantes do que o licenciamento da ISA.

Em uma empresa que desenvolve SoCs próprios, a liberdade de modificar e estender o sistema pode gerar valor estratégico significativo. Em uma equipe pequena que apenas precisa de um MCU confiável e amplamente suportado, um ecossistema proprietário maduro pode reduzir risco e tempo de desenvolvimento. O custo deve ser calculado para o ciclo de vida do produto, e não apenas para a aquisição do componente.

Componente de custo	**Pergunta de engenharia**
Silício/MCU/SoC	qual é o custo por unidade no volume esperado?
Ferramentas	há custos de IDE, debug, análise ou certificação?
Portabilidade	quantas pessoas-mês serão necessárias para migrar firmware?
Validação	quais testes de regressão e conformidade serão exigidos?
Suporte	há SLA, suporte do fornecedor e roadmap de longo prazo?
Manutenção	qual o custo de atualizar toolchain, RTOS, BSP e segurança?

7.6 Ecossistema como resultado de engenharia

A viabilidade técnica não depende apenas da existência de silício. Um ecossistema produtivo inclui compiladores, assemblers, linkers, depuradores, programadores, RTOS, kernels, bibliotecas, HALs, SDKs, exemplos, emuladores, documentação e suporte comunitário ou comercial. A maturidade desse conjunto afeta diretamente o tempo necessário para transformar uma placa em produto.

A disponibilidade de suporte RISC-V em ferramentas amplamente utilizadas reduz a barreira de entrada. GCC, LLVM, GDB, OpenOCD, QEMU e sistemas operacionais/RTOS oferecem diferentes níveis de suporte. Em particular, o Zephyr documenta suporte a RISC-V de 32 e 64 bits e inclui recursos como PMP, user mode e diferentes extensões, demonstrando que o ecossistema de software embarcado já é operacional para uma variedade de plataformas. A avaliação de um projeto, contudo, deve confirmar a qualidade do BSP e do suporte específico ao chip selecionado.

7.7 Síntese dos resultados

A releitura dos resultados conduz a uma conclusão mais precisa que a simples classificação de uma arquitetura como "melhor". RISC-V é tecnicamente viável para hardware embarcado e oferece vantagens estratégicas relevantes em abertura, extensibilidade e liberdade de implementação. Sua adequação concreta depende da disponibilidade de uma implementação que satisfaça os requisitos de desempenho, energia, segurança, confiabilidade, certificação, custo e manutenção do produto.

ARM permanece uma referência importante em sistemas embarcados devido à amplitude de produtos, ferramentas e experiência acumulada. x86, embora menos comum em microcontroladores de baixo consumo, continua relevante em sistemas embarcados de maior capacidade, computação industrial e edge computing. A decisão correta não é ideológica; deve ser rastreável a requisitos e evidências de engenharia.

8 DISCUSSÃO: DA COMPARAÇÃO DE ARQUITETURAS À DECISÃO DE ENGENHARIA

A discussão amplia o estudo comparativo para responder à questão central sob uma perspectiva de decisão tecnológica. A pergunta "RISC-V é viável?" precisa ser complementada por "para qual classe de aplicação, com qual implementação, sob quais requisitos e em qual horizonte de manutenção?". Essa formulação reduz generalizações e aproxima o estudo do processo real de seleção de plataformas.

8.1 ISA, microarquitetura e produto: três conceitos que não devem ser confundidos

Uma ISA define o contrato visível ao software: instruções, registradores, comportamento arquitetural e, conforme o caso, extensões e mecanismos de privilégio. A microarquitetura define como esse contrato é realizado internamente. O produto, por sua vez, integra núcleo, caches, barramentos, memória, periféricos, relógios, energia, debug e recursos de segurança. Comparar "RISC-V versus ARM" sem especificar esses níveis é equivalente a comparar categorias de projeto sem identificar os produtos efetivamente medidos.

A consequência prática é que resultados de um núcleo RISC-V pequeno não devem ser extrapolados para todos os núcleos RISC-V, assim como dados de um Cortex-M não representam todos os produtos ARM. Uma análise rigorosa usa pares comparáveis: mesma

classe de aplicação, faixa de desempenho, tecnologia de memória, conjunto de periféricos e restrições térmicas semelhantes.

8.2 Modularidade da RISC-V como ferramenta de projeto

A RISC-V foi concebida com bases e extensões padronizadas que permitem compor diferentes perfis de arquitetura. Em um microcontrolador, pode ser suficiente uma configuração de 32 bits com extensões selecionadas para multiplicação, instruções compactadas ou operações específicas. Em aplicações Linux e edge, configurações de 64 bits, MMU e extensões adicionais passam a ser relevantes. Essa modularidade permite reduzir funcionalidades desnecessárias ou adicionar recursos voltados à carga de trabalho.

A vantagem da modularidade vem acompanhada de responsabilidade de configuração. Quanto maior a liberdade de composição, maior a necessidade de documentar a string de arquitetura, ABI, extensões obrigatórias, comportamento de exceções e requisitos de toolchain. Para produtos mantidos por muitos anos, uma configuração estável e bem especificada costuma ser mais valiosa do que explorar cada nova extensão assim que ela surge.

8.3 O estado das especificações e a importância de versões ratificadas

A RISC-V International mantém uma biblioteca de especificações ratificadas. Em janeiro de 2026, a biblioteca de referência apresenta versões atualizadas das especificações Unprivileged e Privileged. Para projetos comerciais e acadêmicos, a recomendação é vincular requisitos a versões ratificadas e registrar explicitamente quais

extensões são exigidas. Isso melhora interoperabilidade e reduz o risco de depender de propostas experimentais que mudem durante o ciclo de desenvolvimento.

A mesma disciplina deve ser aplicada ao software. Compiladores e assemblers podem aceitar extensões em ritmos diferentes; SDKs de fabricantes podem introduzir opções próprias; bibliotecas podem exigir ABIs específicas. Um projeto reproduzível registra versões e flags de construção e utiliza integração contínua para detectar regressões de toolchain.

8.4 Toolchain, ABI e reprodutibilidade

A portabilidade de código C ou C++ para RISC-V é frequentemente simples no nível da linguagem, mas o comportamento do firmware também depende de startup, linker script, ABI, intrínsecos, assembly, drivers, acesso a registradores, atomics e convenções do SDK. A migração deve começar pela separação entre lógica de aplicação e dependências da plataforma.

No GCC, opções específicas de RISC-V permitem definir arquitetura e ABI, entre outros aspectos. O mesmo projeto pode gerar binários diferentes ao alterar extensões, estratégia de alinhamento e otimizações. Por isso, comparações de tamanho e desempenho precisam usar builds reproduzíveis, com comandos registrados e artefatos versionados.

Item a registrar	Exemplo de informação	Por que importa

Item a registrar	Exemplo de informação	Por que importa
Compilador	nome e versão	otimizações e geração de código mudam
Arquitetura	RV32/RV64 e extensões	define instruções disponíveis
ABI	convenção adotada	afeta compatibilidade binária
Otimização	-O0/-O2/-Os etc.	muda desempenho e tamanho
Linker script	mapa de memória	impacta flash, RAM e startup
Bibliotecas	libc/RTOS/SDK	podem dominar comportamento do sistema

8.5 Sistema operacional e RTOS

Em dispositivos muito simples, bare metal continua apropriado. À medida que cresce a complexidade, um RTOS pode fornecer escalonamento, sincronização, timers, drivers e abstrações que reduzem o custo de desenvolvimento. RISC-V é suportado por ecossistemas de RTOS e sistemas operacionais, mas a maturidade prática deve ser verificada para a placa escolhida.

O Zephyr é um exemplo relevante porque documenta suporte atual a RISC-V de 32 e 64 bits, além de recursos relacionados a proteção de memória e modo de usuário em plataformas compatíveis. Para uma equipe, isso permite prototipar drivers e aplicações usando APIs portáveis. Entretanto, a existência do suporte arquitetural não elimina

a necessidade de avaliar clock control, pinctrl, DMA, low-power, networking e dispositivos específicos do SoC.

8.6 Segurança por projeto: mecanismos, configuração e ciclo de atualização

A segurança de um dispositivo embarcado é resultado de arquitetura, implementação e processo. Modos de privilégio e mecanismos de proteção de memória fornecem fundamentos, mas a segurança efetiva depende de boot verificado, gestão de chaves, atualização assinada, isolamento de componentes, redução de superfície de ataque e resposta a vulnerabilidades. A plataforma deve ser analisada como cadeia de confiança.

Em RISC-V, a abertura das especificações facilita auditoria e entendimento do comportamento arquitetural. Para alguns produtos, a possibilidade de integrar extensões ou mecanismos próprios pode ser estratégica. Ao mesmo tempo, extensões proprietárias aumentam o custo de manutenção e podem reduzir portabilidade. A recomendação é manter a maior parte possível do software em interfaces padronizadas e encapsular recursos específicos atrás de camadas bem definidas.

8.7 Aplicações de baixo consumo e IoT

RISC-V é particularmente atraente em sistemas nos quais customização, custo de silício, integração de aceleradores e controle sobre o conjunto de instruções são relevantes. Sensores, controladores, dispositivos IoT e sistemas especializados podem se beneficiar de

núcleos pequenos e de extensões selecionadas. Contudo, para um produto alimentado por bateria, a eficiência do sistema de energia, rádio, sensores e firmware pode ter impacto maior que a ISA.

A avaliação deve usar um perfil de missão. Por exemplo: permanecer em sleep por 59 segundos, acordar, adquirir sensores, processar, transmitir e retornar ao sleep. Medir esse ciclo completo fornece informação muito mais útil que comparar apenas valores de potência ativa divulgados em datasheets.

8.8 Edge AI e extensões especializadas

Aplicações de inteligência artificial na borda evidenciam o valor de especialização. Operações vetoriais, DSP, matrizes e aceleradores dedicados podem transformar o perfil de desempenho. RISC-V permite que implementadores combinem extensões padronizadas com aceleradores e interfaces específicas. Nesse contexto, a pergunta deixa de ser somente quantas instruções por segundo o núcleo executa e passa a incluir movimentação de dados, largura de banda de memória, quantização, consumo por inferência e suporte do compilador ao acelerador.

O risco técnico é criar uma solução eficiente no hardware, mas difícil de programar. Uma extensão só produz valor quando a toolchain consegue utilizá-la e quando bibliotecas ou intrínsecos permitem explorar o recurso sem tornar o software impossível de manter. Assim, arquitetura e ferramentas devem evoluir como sistema integrado.

8.9 Cadeia de suprimentos, longevidade e risco de fornecedor

Um dos argumentos estratégicos associados a padrões abertos é a redução da dependência de um único proprietário da ISA. Entretanto, um produto ainda pode depender de um fabricante específico de SoC, de um SDK ou de IPs fechados. A avaliação de risco deve mapear quais camadas são substituíveis e quais criam lock-in.

Para produtos com vida útil longa, devem ser analisados disponibilidade do componente, política de EOL, roadmap, erratas, suporte a segurança, disponibilidade de ferramentas e possibilidade de segunda fonte. A abertura da ISA amplia opções, mas a resiliência da cadeia depende de decisões de produto e arquitetura de software.

8.10 Estratégia experimental recomendada

Para transformar a análise bibliográfica em evidência experimental, propõe-se uma bancada comparativa com pelo menos uma plataforma RISC-V e uma plataforma ARM da mesma classe. O objetivo não é provar superioridade universal, mas construir um conjunto de medições reproduzíveis para uma aplicação-alvo.

- Definir uma carga de trabalho real e um conjunto pequeno de microbenchmarks de apoio.
- Fixar tensão, frequência, modo de compilação e versões de toolchain.
- Registrar consumo em sleep, ativo e durante o ciclo completo da aplicação.

- Medir tempo de execução, latência de interrupção, uso de flash e RAM.
- Executar múltiplas repetições e reportar média, mediana, dispersão e pior caso quando relevante.
- Publicar scripts de build, código do benchmark e arquivos brutos para permitir reprodução.
- Separar conclusões da plataforma medida de inferências mais gerais sobre a arquitetura.

Esse desenho experimental pode ser incorporado em edições futuras do livro e também servir como base para artigo científico ou repositório de dados. Ele responde diretamente à principal limitação do estudo original: a dependência de evidências secundárias e a dificuldade de comparar resultados coletados sob condições heterogêneas.

8.11 Critérios de decisão por classe de produto

Classe de produto	**Prioridades típicas**	**O que validar em RISC-V**
Sensor/IoT	energia, custo, conectividade	sleep, wake-up, rádio, SDK e segurança
Controle industrial	determinismo, robustez, longo ciclo de vida	latências, watchdogs, certificações e suporte
Dispositivo médico	segurança, rastreabilidade, confiabilidade	processo de desenvolvimento e evidências de conformidade
Edge AI	desempenho/W, memória, aceleradores	toolchain, kernels otimizados e energia por inferência

Classe de produto	Prioridades típicas	O que validar em RISC-V
Gateway/Linux	MMU, rede, software, atualização	perfil ISA, kernel, drivers, virtualização/isolamento
SoC customizado	diferenciação, IP, custo de silício	verificação, extensões, ecossistema e manutenção

O quadro mostra que a mesma ISA pode ser excelente em uma classe de produto e inadequada em outra quando a implementação não oferece os recursos necessários. A análise de viabilidade deve, portanto, ser sempre contextual.

8.12 Síntese da discussão

A principal conclusão da discussão é que RISC-V deve ser tratada como uma plataforma arquitetural aberta sobre a qual diferentes soluções podem ser construídas. Essa abertura altera o equilíbrio entre padronização e liberdade de projeto. Para organizações capazes de explorar customização e participar do ecossistema, ela pode reduzir barreiras e ampliar opções. Para equipes que dependem de componentes prontos, a qualidade do fornecedor, do SDK e do suporte continua determinante.

Assim, a viabilidade da adoção em hardware embarcado não é uma propriedade binária da ISA. É uma decisão de engenharia baseada em requisitos, medições, risco, custo total e capacidade organizacional. Essa formulação é mais compatível com a diversidade atual do ecossistema RISC-V e fornece uma base mais sólida para os capítulos

seguintes sobre segurança, aplicações, migração e seleção de plataforma.

9 SEGURANÇA, PRIVILÉGIO E ISOLAMENTO EM PLATAFORMAS RISC-V

Segurança de sistemas embarcados é uma propriedade do sistema completo. A ISA pode fornecer mecanismos úteis, mas não elimina a necessidade de boot seguro, armazenamento de chaves, atualização autenticada, separação de privilégios e proteção física. Em RISC-V, o caráter aberto da especificação facilita inspeção e implementação, porém a segurança efetiva depende do SoC e do firmware.

Um mecanismo importante em muitas implementações é a proteção de memória física, que pode limitar regiões acessíveis por software em diferentes contextos. Em sistemas com suporte a níveis de privilégio, esse mecanismo pode ajudar a separar kernel, aplicações e componentes confiáveis. Em microcontroladores, a disponibilidade e a granularidade desse recurso devem ser verificadas na implementação concreta.

Boot seguro é frequentemente realizado por uma raiz de confiança em ROM que valida o próximo estágio antes da execução. Esse processo pode incluir assinaturas digitais, anti-rollback e proteção de chaves. Embora existam práticas comuns, a presença de RISC-V no núcleo não garante que o SoC ofereça esses recursos. A seleção deve examinar documentação de segurança do fabricante e o modelo de atualização durante todo o ciclo de vida.

Criptografia pode ser implementada em software, por instruções especializadas ou por aceleradores dedicados. Em sistemas restritos, aceleração pode reduzir tempo e energia de operações como autenticação e comunicação segura. Contudo, desempenho criptográfico não deve ser confundido com segurança geral. Implementações precisam considerar geração de entropia, gerenciamento de chaves, resistência a falhas e ataques de canal lateral quando o nível de ameaça justificar.

Abertura do padrão também pode melhorar auditabilidade. Pesquisadores e fornecedores podem analisar especificações sem depender de documentação confidencial. Ainda assim, transparência da ISA não significa transparência de todos os componentes do chip. Um SoC pode conter blocos proprietários, ROMs fechadas e firmware de terceiros. O modelo de confiança deve listar todos esses componentes.

Para aplicações reguladas ou críticas, a decisão deve partir de requisitos de segurança e certificação. Uma implementação nova pode oferecer excelentes mecanismos técnicos e ainda não possuir evidências, ferramentas qualificadas ou histórico suficientes para determinado processo de certificação. Nesses cenários, maturidade e documentação são fatores de primeira ordem.

10 RISC-V EM APLICAÇÕES EMBARCADAS E DE BORDA

O espaço de sistemas embarcados é heterogêneo, e o RISC-V pode ocupar papéis distintos. Em microcontroladores de baixo custo, a atratividade está em núcleos compactos, possibilidade de customização e integração com periféricos específicos. Em plataformas de borda mais potentes, extensões e aceleradores podem atender cargas de processamento de sinais, visão computacional e aprendizado de máquina.

Em IoT, uma plataforma precisa equilibrar consumo, conectividade, segurança e custo. O núcleo de CPU é apenas parte do orçamento energético; rádio, sensores e periféricos podem dominar o consumo. A vantagem de uma implementação RISC-V aparece quando a arquitetura permite adaptar o processamento à carga e quando o ecossistema fornece drivers e protocolos necessários. A análise deve ser realizada no nível do dispositivo completo.

Em automação industrial, requisitos de determinismo, interfaces de campo, temperatura e longevidade ganham peso. A possibilidade de customização pode favorecer controladores especializados, mas fabricantes e integradores precisam garantir manutenção por muitos anos. O roadmap do fornecedor e a disponibilidade de componentes são tão importantes quanto o custo inicial.

Em aplicações automotivas e médicas, a discussão muda de escala devido a requisitos de segurança funcional e processos regulatórios. A adoção é possível apenas quando a implementação e a cadeia de

desenvolvimento fornecem evidências adequadas. A abertura da ISA pode facilitar inovação, mas não substitui qualificação, diagnóstico, redundância e documentação de segurança.

Em processamento de borda, o valor do RISC-V cresce quando extensões ou aceleradores específicos reduzem movimentação de dados. Cargas de IA frequentemente são limitadas por memória e multiplicações matriciais, e um núcleo generalista pode não ser suficiente. A arquitetura aberta permite integração mais estreita entre CPU e aceleradores, criando oportunidades de co-design hardware/software.

Para ensino e pesquisa, RISC-V possui valor adicional por permitir estudar a interface completa sem restrições de acesso à especificação. Esse aspecto pode acelerar formação de engenheiros e prototipagem de novas ideias. Em empresas, entretanto, a decisão deve converter essa flexibilidade em métricas de produto: menor energia, menor custo, melhor desempenho ou menor dependência de fornecedor.

11 ESTRATÉGIA DE ADOÇÃO E MIGRAÇÃO PARA RISC-V

Migrar uma base de software embarcado para RISC-V deve ser tratado como projeto de plataforma. O primeiro passo é inventariar dependências da arquitetura atual: código assembly, intrínsecos, bibliotecas binárias, drivers, bootloader, RTOS, ferramentas de teste e interfaces com depuradores. Quanto maior a dependência de elementos específicos, maior o custo de migração.

O segundo passo é definir um alvo de prova de conceito. A equipe deve selecionar uma placa ou SoC representativo e portar um subconjunto do produto que exercite interrupções, temporização, comunicação, armazenamento e mecanismos de segurança. A prova de conceito precisa produzir medidas, não apenas demonstrar que o código compila. Tempo de boot, consumo, memória, latência e estabilidade devem ser registrados.

O terceiro passo é estabelecer uma configuração reprodutível da toolchain. Versões do compilador, linker, bibliotecas e scripts de build devem ser fixadas. A configuração de ISA e ABI deve estar explícita. Isso reduz problemas de compatibilidade entre máquinas de desenvolvimento e permite comparar resultados ao longo do projeto.

O quarto passo é atacar portabilidade. Código C/C++ deve evitar pressupostos sobre tamanho de tipos, alinhamento, endianness e comportamento indefinido. Camadas de abstração para periféricos e RTOS reduzem dependência de fornecedor. Trechos assembly devem

ser isolados e acompanhados por testes. Essa melhoria beneficia o produto mesmo que a migração seja interrompida.

O quinto passo é executar validação de hardware. Benchmarks sintéticos são úteis para caracterização, mas a decisão deve utilizar cargas reais do produto. Medições de energia precisam ser feitas em condições controladas e incluir estados de sono. Testes temporais devem considerar pior caso. Sistemas críticos exigem planos adicionais de fault injection, watchdog, recuperação e atualização.

O sexto passo é avaliar aspectos econômicos. A equipe deve comparar BOM, licenças, custo de ferramentas, tempo de engenharia, necessidade de treinamento, suporte e risco de fornecimento. Uma plataforma com componente mais barato pode ter custo total maior se exigir meses adicionais de portabilidade ou manutenção.

A migração deve terminar com um gate de decisão. Critérios de aprovação precisam ser definidos antes dos testes para evitar viés. Caso o RISC-V não satisfaça requisitos nesta geração, os resultados ainda são valiosos: a organização passa a conhecer o gap técnico e pode reavaliar futuras plataformas com menor custo.

12 MATRIZ DE DECISÃO PARA SELEÇÃO DE ARQUITETURA

Uma matriz de decisão ajuda a transformar preferências em critérios auditáveis. O processo começa pela atribuição de pesos aos requisitos do produto. Um dispositivo alimentado por bateria pode atribuir peso alto a energia; um controlador industrial pode priorizar confiabilidade e suporte de longo prazo; um produto de alto volume pode tornar custo e liberdade de integração decisivos.

Os critérios recomendados são: desempenho na carga real; energia por tarefa; latência e determinismo; footprint de memória; recursos de segurança; qualidade de depuração; maturidade de drivers; suporte de RTOS ou sistema operacional; disponibilidade de bibliotecas; custo do componente; licenças e royalties; longevidade; cadeia de suprimentos; documentação; certificações; capacidade de customização; risco de dependência de fornecedor.

Cada plataforma candidata pode receber uma nota baseada em evidência. Notas não devem ser atribuídas apenas por marketing. Sempre que possível, a evidência deve ser um teste reproduzível, documento do fabricante, relatório de qualificação ou resultado de prova de conceito. Critérios sem evidência podem ser marcados como risco em aberto.

RISC-V tende a obter pontuação elevada em abertura da ISA, possibilidade de customização e diversidade potencial de fornecedores. O resultado em desempenho, energia, segurança e suporte varia de implementação para implementação. ARM tende a se

beneficiar de um ecossistema historicamente amplo em muitos segmentos embarcados, mas as condições comerciais e de customização dependem do produto e do modelo de licenciamento. x86 permanece relevante em sistemas embarcados de maior capacidade, especialmente quando compatibilidade com software de PC/servidor é um requisito.

A matriz não deve ser usada para produzir uma falsa precisão matemática. Pesos e notas são instrumentos de transparência. O valor principal é obrigar a equipe a registrar por que uma plataforma foi escolhida e quais riscos foram aceitos. Essa documentação facilita revisão de arquitetura, negociação com fornecedores e reavaliações futuras.

Uma boa decisão também inclui critérios eliminatórios. Se uma plataforma não possui faixa de temperatura requerida, não suporta a memória necessária ou não fornece um mecanismo de segurança obrigatório, ela deve ser excluída independentemente da pontuação em outros itens. A engenharia deve distinguir requisitos obrigatórios de características desejáveis.

13 LIMITAÇÕES, RISCOS E PERSPECTIVAS DE EVOLUÇÃO

O principal risco de uma análise sobre RISC-V é generalizar características de implementações particulares. A abertura da ISA favorece diversidade, e diversidade produz variação. Resultados de um núcleo acadêmico, um microcontrolador comercial e um processador de aplicação não são intercambiáveis. Comparações futuras devem identificar claramente modelo, frequência, memória, ferramenta, flags de compilação e carga de trabalho.

Outro risco é confundir maturidade da especificação com maturidade do ecossistema de um produto. Uma extensão pode estar padronizada e ainda possuir suporte desigual em compiladores, depuradores ou sistemas operacionais. Da mesma forma, um fabricante pode oferecer uma plataforma madura mesmo quando determinada área do ecossistema geral ainda está evoluindo. O engenheiro precisa avaliar a cadeia específica que será usada no produto.

O ritmo de evolução do RISC-V é simultaneamente força e desafio. Novas extensões e perfis ampliam capacidade e melhoram padronização, mas equipes precisam controlar versões para garantir reprodutibilidade. Em produtos de ciclo longo, atualizações de toolchain e especificações devem ser planejadas, não adotadas automaticamente.

A tendência de especialização de hardware favorece arquiteturas modulares. Processamento de IA, criptografia, DSP e domínios industriais podem se beneficiar de extensões e aceleradores próximos

ao núcleo. O valor do RISC-V pode crescer justamente como ponto de integração entre software portável e hardware especializado. O desafio será preservar compatibilidade suficiente para evitar fragmentação excessiva.

Para o Brasil e para ambientes acadêmicos, a arquitetura também possui relevância estratégica pela possibilidade de formação, pesquisa e desenvolvimento de propriedade intelectual sobre uma especificação aberta. Isso não elimina dependências de ferramentas EDA, fabricação e cadeia global de semicondutores, mas amplia o espaço de atuação em projeto de processadores, SoCs, firmware e sistemas embarcados.

Este livro possui limitações decorrentes de sua origem bibliográfica. Parte dos dados quantitativos foi compilada no estudo de 2024 e deve ser atualizada quando uma edição futura incorporar experimentos controlados em hardware contemporâneo. Uma evolução natural da pesquisa é construir uma bancada de benchmark com placas ARM e RISC-V equivalentes, medição elétrica instrumentada e workloads abertos, permitindo replicação independente.

14 BENCHMARKS E MEDIÇÕES PARA SISTEMAS EMBARCADOS

Benchmark é útil apenas quando representa a decisão que precisa ser tomada. Em sistemas embarcados, uma única pontuação agregada pode esconder comportamentos críticos, como latência de interrupção, gargalos de memória ou consumo durante picos. A avaliação deve combinar microbenchmarks, cargas representativas e medições do produto. Microbenchmarks isolam operações; cargas representativas aproximam o uso real; testes de sistema revelam interação entre CPU, memória, barramentos e periféricos.

Para comparar plataformas RISC-V e ARM, a primeira regra é controlar as condições. Frequência de clock, compilador, nível de otimização, memória, caches e periféricos devem ser documentados. Se os chips utilizarem tecnologias de fabricação diferentes, a comparação deve reconhecer esse fator. O objetivo não é provar superioridade abstrata de uma ISA, mas determinar qual implementação atende melhor ao requisito do produto.

O desempenho pode ser medido por tempo total de uma tarefa, throughput, latência e utilização de CPU. Em firmware de controle, o tempo de resposta de uma interrupção pode ser mais relevante que operações por segundo. Em processamento de sinais, ciclos por amostra ou por bloco podem ser adequados. Em comunicação, deve-se medir taxa sustentada e custo de processamento por pacote. A métrica precisa corresponder à função do sistema.

Medição energética exige cuidado adicional. Potência instantânea, potência média e energia por tarefa são grandezas diferentes. Um processador pode apresentar potência maior durante alguns milissegundos e ainda consumir menos energia se concluir a tarefa rapidamente e retornar ao sono. Para dispositivos com duty cycle baixo, é necessário medir também corrente de repouso, tempo de entrada e saída dos modos de sono e consumo dos periféricos que permanecem ativos.

Os instrumentos e o método devem ser registrados. Uma medição feita apenas pela leitura de um monitor interno de corrente pode ter resolução insuficiente para pulsos rápidos. Em protótipos, shunt e instrumento apropriado permitem capturar forma de onda; em fases posteriores, analisadores de potência podem melhorar precisão. O importante é que os resultados sejam reproduzíveis e tenham incerteza compatível com a diferença que se pretende demonstrar.

Benchmarks de memória devem distinguir largura de banda e latência. Núcleos simples podem ser limitados por flash, SRAM externa ou barramento. Caches alteram substancialmente os resultados, assim como prefetch e alinhamento. Para código executado diretamente de memória não volátil, wait states podem dominar o tempo de execução. Esses fatores pertencem à plataforma e não podem ser atribuídos somente à ISA.

A análise deve incluir tamanho de código e dados. Em microcontroladores de baixo custo, alguns kilobytes podem alterar a

escolha do componente. Compare o mesmo firmware com flags equivalentes e separe bibliotecas não utilizadas. Quando extensões comprimidas ou instruções especializadas estão presentes, avalie se reduzem de fato o binário e se o ganho compensa eventual aumento de complexidade da plataforma.

Por fim, resultados devem ser apresentados com contexto, não apenas como ranking. Uma tabela útil informa hardware, frequência, memória, toolchain, flags, versão do firmware, condições de energia e número de repetições. Essa prática torna o estudo auditável e evita conclusões genéricas a partir de números que dependem de configurações particulares.

15 PORTABILIDADE DE FIRMWARE E QUALIDADE DE SOFTWARE

A portabilidade de firmware é um dos custos mais subestimados em mudanças de arquitetura. Projetos maduros acumulam dependências de registradores, intrínsecos, código assembly, drivers e bibliotecas fornecidas pelo fabricante. Migrar para RISC-V pode ser simples em aplicações bem modularizadas e difícil em produtos fortemente acoplados ao microcontrolador anterior. Antes da decisão, o código deve ser classificado por nível de dependência de hardware.

Uma arquitetura de software em camadas reduz esse custo. A lógica de negócio e algoritmos devem permanecer independentes do SoC sempre que possível. A camada de abstração de hardware concentra acesso a GPIO, timers, ADC, comunicação e outros periféricos. O RTOS pode oferecer interfaces portáveis para tarefas, mutexes e filas. Essa separação não elimina diferenças de hardware, mas torna explícito onde a migração exige trabalho.

C e C++ também contêm armadilhas de portabilidade. Pressupor tamanho de int, alinhamento de estruturas ou comportamento de casts pode gerar falhas ao mudar de compilador e ABI. Código que depende de comportamento indefinido pode funcionar por anos em uma plataforma e falhar em outra. A migração é uma oportunidade para ativar warnings, sanitização em testes de host e análise estática, elevando qualidade geral do firmware.

Código assembly merece tratamento especial. Rotinas de inicialização, troca de contexto, acesso a instruções especiais e otimizações manuais precisam ser portadas. O primeiro passo é localizar todos os trechos assembly e intrínsecos específicos. Em seguida, deve-se decidir se continuam necessários. Compiladores modernos podem produzir código adequado para muitas rotinas que no passado eram otimizadas manualmente.

Bibliotecas binárias fechadas são um risco de bloqueio. Se um produto depende de middleware distribuído apenas para ARM, a migração para RISC-V pode exigir substituição ou novo acordo com fornecedor. Criptografia, codecs, stacks de comunicação e bibliotecas de sensores devem ser inventariados antes de qualquer prova de conceito. A existência de código-fonte ou suporte multiplataforma reduz significativamente o risco.

Os testes automatizados são o principal mecanismo para preservar comportamento durante a migração. Funções independentes de hardware podem ser testadas no host. Interfaces de hardware podem utilizar mocks. Emuladores podem cobrir parte do caminho, enquanto testes hardware-in-the-loop validam temporização, periféricos e interação física. A meta é separar defeitos de portabilidade de defeitos da nova plataforma.

Outra questão é a atualização em campo. Se o formato de imagem, bootloader ou esquema de partição mudar, deve existir estratégia de transição. Produtos instalados não podem ser tratados como placas de

laboratório. Compatibilidade de protocolo, dados persistentes e mecanismos de rollback precisam ser preservados. Uma migração de arquitetura bem executada mantém interfaces externas estáveis mesmo quando o hardware interno muda.

Portabilidade não significa buscar o menor denominador comum. Recursos específicos do RISC-V podem ser usados quando oferecem vantagem mensurável. O importante é encapsular tais dependências e fornecer caminhos alternativos quando necessário. Dessa forma, a organização aproveita a plataforma sem transformar toda a base de código em software impossível de mover novamente.

16 SELEÇÃO DE NÚCLEO, SOC E PLATAFORMA

Escolher RISC-V é apenas o início da seleção. O produto real utiliza um núcleo implementado dentro de um SoC, acompanhado de memória, clocks, controladores de interrupção, interfaces, aceleradores e blocos de segurança. Dois SoCs RISC-V podem ser menos semelhantes entre si, do ponto de vista do firmware, do que duas famílias de microcontroladores baseadas em ISAs diferentes. Por isso, a unidade de comparação deve ser a plataforma completa.

O núcleo deve ser caracterizado por largura de dados, pipeline, caches, unidade de ponto flutuante, extensões implementadas e recursos de privilégio. Para controle simples, uma implementação pequena pode oferecer melhor relação custo-energia. Para processamento de dados, caches e extensões adicionais podem ser decisivos. O maior conjunto de recursos não é necessariamente melhor quando aumenta consumo e área sem benefício para a carga.

O subsistema de memória frequentemente determina o desempenho. É necessário analisar quantidade e tipo de SRAM, flash interna ou externa, largura dos barramentos, DMA, caches e proteção. Firmware em tempo real pode exigir memória com latência previsível. Aplicações com grandes modelos ou buffers precisam de capacidade e largura de banda. A CPU não compensa uma arquitetura de memória inadequada.

Periféricos devem ser avaliados segundo necessidades do produto, não pela quantidade presente no datasheet. Interfaces de comunicação,

timers, PWM, ADC, DAC, USB, Ethernet e controladores de memória precisam ter desempenho e características compatíveis. Mais importante: deve existir documentação suficiente e drivers estáveis. Um periférico avançado sem driver confiável pode aumentar o cronograma em semanas.

O sistema de interrupções e temporizadores merece teste específico. Sistemas de tempo real dependem da forma como prioridades, mascaramento e encaminhamento são implementados. Não basta assumir que todo RISC-V possui comportamento idêntico nessa área. O SoC pode adotar controladores e topologias diferentes. A documentação e o RTOS precisam estar alinhados com a implementação.

Recursos de segurança da plataforma incluem mais que instruções. Verifique ROM de boot, autenticação de imagem, armazenamento protegido de chaves, gerador de números aleatórios, aceleração criptográfica, debug lock, proteção de memória e mecanismos de atualização. Em produtos conectados, a ausência de um componente pode exigir implementação externa e eliminar uma aparente economia do chip.

A cadeia de suprimentos também faz parte da plataforma. Avalie fabricante, distribuidores, lead time, política de longevidade, PCN e alternativas compatíveis. Uma ISA aberta aumenta possibilidades de fornecedores no ecossistema, mas isso não garante substituição pin-to-

pin ou software totalmente compatível. Diversificação precisa ser planejada no projeto elétrico e no software.

Finalmente, a plataforma deve ser avaliada pelo suporte de engenharia. Documentação clara, exemplos, erratas públicas, fóruns técnicos, suporte comercial e disponibilidade de placas de desenvolvimento reduzem risco. Para equipes pequenas, esses fatores podem valer mais que diferenças marginais no preço unitário. A melhor plataforma é aquela cujo conjunto de hardware, software e suporte atende ao ciclo de vida do produto.

17 PADRÃO ABERTO, HARDWARE ABERTO E MODELOS DE LICENCIAMENTO

Uma fonte recorrente de confusão é tratar RISC-V como sinônimo de hardware open source. RISC-V é uma ISA aberta e padronizada. Uma empresa pode implementar essa ISA em um núcleo totalmente proprietário, em um núcleo disponibilizado sob licença aberta ou em uma combinação de componentes. Para avaliar custo e liberdade tecnológica, é necessário identificar a licença de cada camada.

O padrão aberto permite implementar a interface arquitetural sem negociar uma licença de ISA nos mesmos moldes de arquiteturas proprietárias. Essa característica reduz uma barreira, mas não elimina custos de propriedade intelectual. Núcleos comerciais, interconexões, controladores, aceleradores, interfaces de alta velocidade e ferramentas EDA podem possuir licenças próprias. Um SoC competitivo normalmente combina múltiplos componentes.

Hardware aberto pode oferecer vantagens educacionais e de auditabilidade. Quando HDL e documentação estão disponíveis, equipes podem estudar microarquitetura, modificar blocos e experimentar extensões. Para produção comercial, entretanto, a licença deve ser analisada quanto a obrigações, patentes, redistribuição e suporte. “Aberto” não significa ausência de condições jurídicas ou de responsabilidades de integração.

O custo de licenciamento precisa ser separado do custo de desenvolvimento. Uma solução sem royalties pode exigir mais

engenharia interna; outra com licenciamento pode incluir suporte, validação e componentes prontos. O cálculo correto compara custo total em função do volume e do tempo. Em produtos de milhões de unidades, pequenas diferenças por componente podem dominar. Em volumes baixos, meses de engenharia podem ser muito mais relevantes.

Outro aspecto é o lock-in. Uma ISA aberta pode reduzir dependência no nível do conjunto de instruções, mas um produto ainda pode ficar dependente de periféricos exclusivos, SDK proprietário, formato de boot ou extensão não padronizada. Portabilidade real exige limitar dependências não essenciais e registrar onde elas são aceitas por vantagem técnica.

Perfis e especificações ratificadas ajudam a criar uma base comum. Quanto maior a convergência entre implementações de uma mesma classe, mais fácil distribuir software e ferramentas. Para sistemas embarcados muito especializados, a liberdade de customização continuará valiosa; para plataformas de aplicação, compatibilidade tende a assumir peso crescente.

Do ponto de vista estratégico, o RISC-V amplia opções de negociação e projeto. Uma organização pode comprar chips prontos, licenciar núcleos, utilizar núcleos abertos ou desenvolver propriedade intelectual própria. Cada caminho possui custo, risco e competência necessária diferentes. A decisão deve estar alinhada ao negócio: uma

empresa de produto talvez não precise projetar CPU; uma empresa de semicondutores pode considerar isso parte de sua diferenciação.

18 ROADMAP DE ADOÇÃO ORGANIZACIONAL

A adoção de uma nova arquitetura não é apenas decisão técnica. Ela afeta competências, processos, fornecedores e manutenção. Um roadmap reduz risco ao dividir a mudança em etapas verificáveis. A primeira etapa é capacitação: a equipe precisa compreender ISA, ABI, toolchain, modelo de interrupções, debug e características do SoC candidato. Treinamento curto e experimentos controlados evitam que o projeto de produto seja usado como ambiente de aprendizagem inicial.

A segunda etapa é laboratório. Uma placa de desenvolvimento deve ser integrada ao fluxo normal da equipe, com repositório, build automatizado, testes e depuração documentada. Pequenos exemplos — GPIO, timer, comunicação, RTOS, armazenamento e segurança — formam uma base de conhecimento interna. O objetivo é transformar informação dispersa em procedimentos repetíveis.

A terceira etapa é portabilidade. Um módulo real, mas não crítico, pode ser migrado. Nessa fase a equipe identifica dependências ocultas e mede esforço. As mudanças de abstração e testes devem retornar para a base principal de software, aumentando portabilidade geral. O trabalho não deve ser tratado como branch experimental descartável se produzir melhorias estruturais.

A quarta etapa é prova de conceito de produto. A carga principal deve executar no hardware candidato, e métricas de desempenho, energia, memória e estabilidade devem ser comparadas ao baseline. Requisitos

eliminatórios são avaliados primeiro. A prova de conceito deve incluir build de produção, mecanismo de atualização e procedimentos de diagnóstico, não apenas uma demonstração funcional.

A quinta etapa é análise de fornecedor e ciclo de vida. Compras, engenharia e gestão de produto devem avaliar disponibilidade, contratos, suporte, erratas e política de longevidade. Para produtos regulados, qualidade e certificação entram no mesmo gate. Essa etapa evita que uma decisão tecnicamente atraente se torne inviável comercialmente.

A sexta etapa é piloto. Uma pequena série permite descobrir variações de fabricação, problemas de programação, testes de fábrica e comportamento de campo. Telemetria de diagnóstico, quando apropriada e respeitando privacidade, pode ajudar a comparar estabilidade. Somente após o piloto a plataforma deve ser promovida a opção preferencial para determinada classe de produto.

O roadmap deve incluir critérios para interromper a migração. Manter uma arquitetura alternativa é racional quando riscos permanecem altos. A abertura do RISC-V significa que o ecossistema continua evoluindo; uma decisão “não agora” pode ser revista com novos componentes e ferramentas. O conhecimento adquirido reduz custo da próxima avaliação.

Por fim, a organização deve manter uma ficha de plataforma. Esse documento registra ISA/extensões, toolchain homologada, versões de SDK, erratas, configurações de segurança, fornecedores, testes de

referência e decisões arquiteturais. A ficha transforma a adoção em ativo institucional e reduz dependência do conhecimento individual de um desenvolvedor.

19 CONCLUSÕES

A pergunta central desta obra não admite uma resposta universal. RISC-V é viável para hardware embarcado quando a implementação selecionada, o ecossistema de software e o processo de engenharia satisfazem os requisitos da aplicação. A arquitetura oferece vantagens concretas de abertura, modularidade e possibilidade de customização, mas essas características não substituem validação de desempenho, energia, confiabilidade e segurança.

O estudo original identificou potencial competitivo principalmente em flexibilidade e custo, ao mesmo tempo em que reconheceu a maturidade das plataformas consolidadas em diferentes aplicações. A edição ampliada reforça essa conclusão e acrescenta que o custo de adoção deve incluir software, ferramentas, treinamento, suporte e risco de ciclo de vida. Em alguns projetos, a economia de licenciamento pode ser decisiva; em outros, o esforço de migração ou a ausência de certificações pode dominar a decisão.

Para aplicações de baixa e média complexidade, prototipagem, ensino, IoT e sistemas que valorizam customização, o RISC-V pode constituir uma alternativa especialmente atraente. Para aplicações críticas, a análise deve se deslocar da ISA abstrata para uma plataforma qualificada, com evidências de segurança, mecanismos de diagnóstico e suporte compatível com o setor.

A contribuição mais importante do RISC-V para a engenharia embarcada talvez seja ampliar o espaço de projeto. Em vez de aceitar

uma ISA como elemento fixo e fechado, equipes podem considerar a interface entre hardware e software como parte da arquitetura do produto. Essa liberdade aumenta possibilidades de inovação, mas exige práticas rigorosas de especificação, teste e governança técnica.

Como recomendação final, a adoção deve começar por prova de conceito e matriz de decisão, avançar para medições com cargas reais e terminar com revisão de riscos de ciclo de vida. A escolha de arquitetura deve ser consequência de evidências. Nessa perspectiva, RISC-V não é apenas uma alternativa às arquiteturas existentes, mas uma plataforma que incentiva uma forma mais aberta e explícita de projetar sistemas computacionais.

APÊNDICE A – ROTEIRO DE AVALIAÇÃO TÉCNICA

Este roteiro pode ser utilizado em uma prova de conceito para comparar plataformas. O objetivo é produzir um dossiê de evidências antes da decisão de arquitetura. Para cada item, registre a versão do hardware, firmware, compilador, configuração de ISA/ABI e condições de teste.

1. Definir requisitos obrigatórios: memória, interfaces, tensão, temperatura, tempo real, segurança e certificação.
2. Selecionar pelo menos duas plataformas candidatas e documentar a configuração de CPU, memória e periféricos.
3. Fixar versões de compilador, linker, bibliotecas e sistema operacional/RTOS.
4. Compilar a mesma carga de trabalho com opções equivalentes e registrar tamanho do binário.
5. Medir tempo de execução e latência das funções críticas.
6. Medir corrente e energia por tarefa em execução, idle e sono.
7. Testar tempo de boot, reset, watchdog e recuperação após falha.
8. Validar depuração: flash, breakpoint, watchpoint, inspeção de registradores e exceções.
9. Validar atualização de firmware e mecanismos de autenticação/anti-rollback, quando aplicável.
10. Executar testes de estabilidade prolongada e registrar falhas.
11. Levantar custo de BOM, licenças, ferramentas, suporte e treinamento.
12. Levantar disponibilidade e política de longevidade do fornecedor.

13. Preencher matriz de decisão com evidências e registrar riscos ainda não mitigados.
14. Realizar revisão técnica independente antes da aprovação para produto.

APÊNDICE B – GLOSSÁRIO ESSENCIAL

ABI: Application Binary Interface. Conjunto de convenções que permite interoperabilidade binária entre programas, bibliotecas e sistema operacional.

Bare metal: Software executado diretamente sobre o hardware, sem sistema operacional completo.

Bootloader: Software responsável por iniciar a plataforma e carregar ou validar o firmware principal.

CISC: Complex Instruction Set Computing. Designação histórica para arquiteturas com conjuntos de instruções mais complexos.

CPI: Cycles per Instruction. Número médio de ciclos de clock por instrução em determinada carga.

CSR: Control and Status Register. Registrador usado para controle e estado arquitetural.

Debug: Conjunto de técnicas e interfaces para observar e controlar a execução de software durante desenvolvimento e diagnóstico.

DSP: Digital Signal Processing. Processamento digital de sinais e, por extensão, recursos especializados para esse tipo de carga.

ECC: Error-Correcting Code. Técnicas de detecção e correção de erros, frequentemente usadas em memória.

Firmware: Software de baixo nível associado diretamente ao funcionamento de um dispositivo.

ISA: Instruction Set Architecture. Contrato arquitetural entre software e processador.

IoT: Internet of Things. Dispositivos conectados capazes de coletar, processar ou trocar dados.

Microarquitetura: Organização interna usada para implementar uma ISA: pipeline, caches, execução, predição e outros elementos.

PMP: Physical Memory Protection. Mecanismo de proteção de regiões de memória física em implementações RISC-V que o suportam.

RISC: Reduced Instruction Set Computing. Abordagem de arquitetura associada a instruções regulares e operações relativamente simples.

RISC-V: Arquitetura de conjunto de instruções aberta e modular padronizada pela RISC-V International.

RTOS: Real-Time Operating System. Sistema operacional projetado para fornecer mecanismos adequados a requisitos temporais previsíveis.

SoC: System on Chip. Circuito integrado que combina CPU, memória/interface de memória, periféricos e outros blocos em um único chip.

Toolchain: Conjunto de ferramentas de desenvolvimento, como compilador, assembler, linker, debugger e utilitários.

Watchdog: Temporizador ou mecanismo de supervisão usado para detectar travamentos e iniciar recuperação do sistema.

REFERÊNCIAS

ASANOVIĆ, Krste; PATTERSON, David A. The RISC-V Reader: An Open Architecture Atlas. Strawberry Canyon: RISC-V Foundation, 2017.

BARR, Michael; MASSA, Anthony. Programming Embedded Systems: With C and GNU Development Tools. 2. ed. Sebastopol: O'Reilly Media, 2006.

BARROS, Edna; CAVALCANTE, Sérgio. Introdução aos Sistemas Embarcados. Recife: Grupo de Engenharia da Computação – GRECO, Centro de Informática – CIn, Universidade Federal de Pernambuco – UFPE, [2024]. Disponível em: < https://www.cin.ufpe.br/~svc/ese/Introducao%20aos%20Sistemas%20Embarcados.pdf>.

Acesso em: 23 out. 2024.

BROOKS, D.; MARTONOSI, M. Dynamic thermal management for high-performance microprocessors. In: Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA). Monterrey: IEEE, 2001. p. 171-182.

BROOKS, David; MARTONOSI, Margaret. Dynamic thermal management for high-performance microprocessors. In: Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA), 2001, Monterrey, CA. IEEE, 2001. p. 171-182.

CELIO, C. et al. The renewed case for the reduced instruction set computer: avoiding ISA bloat with macro-op fusion for RISC-V. Technical Report No. UCB/EECS-2016-130, Berkeley: University of California, 2016. Disponível em:

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-130.html. Acesso em> 20 out 2024.

COTA, Érika; LUBASZEWSKI, Marcelo. Reliability, Availability and Serviceability of Networks on Chip. New York: Springer, 2011.

FRANÇA, Júnia Lessa; VASCONCELLOS, Ana Cristina de.; MAGALHÃES, Maria Helena de Andrade; BORGES, Stella Maris. Manual para normalização de publicações técnico-científicas. 10. ed. 1a reimpressão. Belo Horizonte: Editora UFMG, 2021. 250 p.

HALFHILL, Tom R.. RISC-V Offers Freedom of Choice. Microprocessor Report, v. 30, n. 5, p. 1-6, 2016.

LEE, Edward A.; SESHIA, Sanjit A.; et al. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. 2. ed. Cambridge: MIT Press, 2020.

MANSUR, Rodolfo Labiapari. Introdução ao Processador Risc-V. Disponível em:

<https://www2.decom.ufop.br/imobilis/o-risc-v/> . Acessado em: 18 set. 2024.

MARWEDEL, P. Embedded system design: embedded systems foundations of cyber-physical systems. 2. ed. Dordrecht: Springer, 2010.

MARWEDEL, Peter. Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems. 2. ed. Dordrecht: Springer, 2010.

MENZGER, Bejamin W. et al. A Survey of the RISC-V Architecture Software Support. in IEEE Access, [S.L.], v.10, p. 51394-51411. Disponível:< https://ieeexplore.ieee.org/document/9771410 >. Acesso em: 30 det. 2024

NUNES, W. A. et al. RS5: an integrated hardware and software ecosystem for RISC-V embedded systems. PUCRS – School of Technology, Porto Alegre. Disponível em:

<https://ieeexplore.ieee.org/document/10506171> Acesso em: 26 out 2024.

PATTERSON, David A.; HENNESSY, John L. Computer organization and design: the hardware/software interface, ARM edition. San Francisco: Elsevier Inc., 2016.

PATTERSON, David A.; HENNESSY, John L. Computer organization and design: MIPS edition. 6. ed. San Francisco: Morgan Kaufmann, 2021.

PATTERSON, David A.; HENNESSY, John L. Computer organization and design RISC-V edition: the hardware/software interface. Cambridge: Morgan Kaufmann, 2017. 696 p.

PATTERSON, David A.; HENNESSY, John L. Organização e projeto de computadores: a interface hardware software. 5. ed. Rio de Janeiro: GEN LTC, 2017. 680 p.

PATTERSON, David A.; HENNESSY, John L.. Computer architecture: a quantitative approach. 6. ed. San Francisco: Morgan Kaufmann, 2019.

SINCLAIR, Bruce; SERRA, Afonso Celso da Cunha. IoT: Como Usar a "Internet Das Coisas" Para Alavancar Seus Negócios. Belo Horizonte: Autêntica Business, 2018. 272p.

SOUZA, Eduardo Michel Deves de Souza. et al. RVSH - Um processador RISC-V para fins didáticos.In: Anais do Computer on the Beach. São José: Universidade do Vale do Itajaí, 2023. Disponível em :

<https://periodicos.univali.br/index.php/acotb/article/view/19488/11294&ved=2ahUKEwjlq9 PClcmFAxXKO7kGHetOAYUQFnoECBAQAQ&usg=AOvVaw3S4_rwBfC0_h9hwvxm27 Kk>. Acesso em: 18 set. 2024

TANENBAUM, Andrew S; AUSTIN, Todd. Organização estruturada de computadores. São Paulo: Person Prentice Hall, 2013.

VAHID, F.; GIVARGIS, T. Embedded system design: a unified hardware/software introduction. 2. ed. Hoboken: John Wiley & Sons, 2011.

WATERMAN, A.; ASANOVIĆ, K. The RISC-V instruction set manual: volume I: user-level ISA, version 2.2. Berkeley: University of California, 2017.

WAZLAWICK, Paul Sidnei. Metodologia de pesquisa para a ciência da computação. 3 º ed. LTC, 2021.

ZHANG, Fan; KOEHLER, Matthew; BARSI, Jesse; PATEL, Yunsup; ASANOVIC, Krste; GUNTHER, Neal. The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2020.6. Berkeley: RISC-V Foundation, 2020.

RISC-V INTERNATIONAL. RISC-V Ratified Specifications Library. Disponível em: docs.riscv.org/reference/home/index.html. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. The RISC-V Instruction Set Manual: Unprivileged Architecture. Versão publicada na biblioteca de especificações ratificadas, 2026. Disponível em: docs.riscv.org. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. The RISC-V Instruction Set Manual: Privileged Architecture. Versão publicada na biblioteca de especificações ratificadas, 2026. Disponível em: docs.riscv.org. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. RVA23 Profile, version 1.0. 2024. Disponível em: docs.riscv.org. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. RISC-V Debug Specification, version 1.0. 2025. Disponível em: docs.riscv.org. Acesso em: 15 ago. 2026.

ZEPHYR PROJECT. Zephyr support status on RISC-V processors. Documentação oficial. Disponível em: docs.zephyrproject.org. Acesso em: 15 ago. 2026.

FREE SOFTWARE FOUNDATION. Using the GNU Compiler Collection: RISC-V Options. Documentação oficial do GCC. Disponível em: gcc.gnu.org/onlinedocs/. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. RISC-V Ratified Specifications Library. Unprivileged ISA e Privileged ISA, versões de referência publicadas em janeiro de 2026. Disponível em: docs.riscv.org. Acesso em: 15 ago. 2026.

RISC-V INTERNATIONAL. Ratified Specifications. Especificações, extensões e documentos de suporte ratificados do padrão RISC-V. Disponível em: riscv.org/specifications/ratified/. Acesso em: 15 ago. 2026.

APÊNDICE C – QUADROS E TABELAS DO ESTUDO ORIGINAL

Os quadros e tabelas a seguir são reproduzidos e normalizados a partir da versão acadêmica de 2024 para preservar a rastreabilidade do estudo que originou esta edição. Recomenda-se validar novamente valores, unidades, referências e condições experimentais antes de utilizar os números como base de decisão de produto.

Quadro C.1 – Comparação entre arquiteturas RISC-V e ARM

Métrica	**RISC-V**	**ARM**
Eficiência energética por operação	0,8 pJ/instrução	0,6 pJ/instrução
Latência de interrupções (ISR)	20 ciclos	15 ciclos
Gerenciamento de energia (DVFS)	Suporte básico	Completo com DVFS
Tempo de inicialização	200 ms	100 ms
Criptografia assimétrica (RSA/ECC)	Dependente de software	Otimizado com hardware dedicado
Operações matemáticas avançadas	Suporte básico	Suporte com NEON/VFPv4
Firmwares seguros	Presente em algumas extensões/implementações	TrustZone e Secure Boot em plataformas compatíveis
Compressão de dados	Limitada em software	Suporte SIMD em plataformas compatíveis
Custo de desenvolvimento de software	Associado à abertura da ISA	Variável conforme plataforma/licenciamento

Taxa de erro em operações matemáticas	Moderada no quadro original	Baixa com ECC no quadro original
Compensação/correção de erro	Básico no quadro original	Suporte ECC no quadro original
Personalização e extensibilidade	Alta modularidade	Extensível com limitações do modelo adotado
Modos de baixa potência	Básico no quadro original	Avançados em plataformas compatíveis
Consumo em standby	0,5 µW	0,3 µW
Resiliência a temperaturas extremas	Moderada	Alta
TinyML	Dependente de extensões	Ecossistema/otimizações amplas no quadro original
Segurança e autenticação de firmware	Básico no quadro original	TrustZone/Secure Boot em plataformas compatíveis

Fonte no estudo original: adaptado pelo autor com base em Lee et al. (2020). Os números e generalizações devem ser revalidados para a implementação concreta.

Tabela C.2 – Consumo de processadores ARM e RISC-V no estudo original

Processador	**Operação (mW/MHz)**	**Inativo (mW)**	**Baixa potência (mW)**
ARM Cortex-M0	0,04	0,1	0,9
ARM Cortex-M3	0,09	0,3	1,2
ARM Cortex-M4	0,12	0,5	1,5
SiFive E31 (RISC-V)	0,06	0,2	1,0
SiFive E51 (RISC-V)	0,10	0,4	1,3

Fonte no estudo original: elaborado pelo autor com base em Zhan et al. (2020).

Quadro C.3 – Confiabilidade entre arquiteturas no estudo original

Métrica	**RISC-V**	**ARM**	**X86**
Resiliência a erros de hardware (ECC)	Parcial/depende de extensões	Sim	SSim
Algoritmos de correção de erros	Limitado	Sim	Sim
Taxa de falha em operações críticas	Não indicado como vantagem	Sim	Sim
Alta temperatura operacional	Não indicado como vantagem	Sim	Sim
Autocorreção de falhas	Não	Sim	Sim
Execuções longas sem reboot	Sim	Sim	Sim

Resiliência contra SEU	Não no quadro original	Sim	Sim
Firmware seguro / trusted boot	Não no quadro original	Sim	Sim
Diagnóstico de falhas avançado	Não no quadro original	Sim	Sim
Modularidade/flexibilidade	Sim	Não	Não
Custo de redundância	Baixo	Não	Não
Suporte de longo prazo	Depende de comunidade/fornecedor	Sim	Sim
Baixo consumo/standby avançado	Não no quadro original	Sim	Não
Temperaturas extremas	Não no quadro original	Sim	Sim
TinyML	Não no quadro original	Sim	Não
Autenticação de firmware completa	Não no quadro original	Sim	Sim

Fonte no estudo original: elaborado pelo autor com base em Cota e Lubaszewski (2011). A edição ampliada ressalta que esses recursos dependem da implementação e não da ISA de forma isolada.

Quadro C.4 – Tipos de licenciamento no estudo original

Arquitetura	**Licenciamento**	**Custo**	**Vantagens**	**Desvantagens**

RISC-V	ISA aberta	Sem royalties pela ISA; outros componentes podem ter custos	Flexibilidade; possibilidade de redução de custos e customização	Maturidade e suporte variam por implementação/ecossistema
ARM	Proprietária/comercial	Custos dependem do modelo de produto/licença	Ecossistema maduro e suporte amplo	Menor liberdade no nível da ISA; custos comerciais variáveis
x86	Proprietária/comercial	Custos incorporados ao produto/plataforma	Ecossistema maduro e alta compatibilidade de software	Menor customização da ISA e maior consumo em algumas classes embarcadas

Fonte no estudo original: elaborado pelo autor com base em Asanović e Patterson (2017) e Patterson e Hennessy (2018).

Sinopse

Este livro analisa a arquitetura RISC-V e discute sua viabilidade como alternativa para projetos de hardware embarcado. A obra compara RISC-V com arquiteturas consolidadas, como ARM e x86, considerando desempenho, eficiência energética, confiabilidade, custo, flexibilidade de projeto e maturidade do ecossistema.

Ao longo dos capítulos, o leitor encontra uma abordagem técnica e aplicada, com fundamentos de arquitetura de computadores, análise comparativa, implicações para sistemas embarcados e reflexões sobre adoção tecnológica. O resultado é uma referência útil para estudantes, pesquisadores e profissionais interessados em inovação em hardware e sistemas embarcados.

Sobre o autor

Alberto José Oliveira Pereira é engenheiro de computação, com interesse em arquitetura de computadores, sistemas embarcados, software e tecnologia da informação.

1010 0011 0101 1101
1101 1001 0010 0101

thesis editora científica

ISBN 978-658319968-3