

Computação Aproximada: Estudo Comparativo entre Instruções Exatas, Aproximadas e Mistas em RISC-V

Eduardo Paes¹, Hytalo Bassi¹, Jailton Nunes Filho¹, Pedro e Silva¹,
Jhullya Martins¹, Daniela Catelan¹

¹ Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (FACOM - UFMS)
Av. Costa e Silva, s/n – Campo Grande – MS – Brasil

{eduardo-paes, hytalo.bassi, jailton.nunes-filho}@ufms.br

{pedro.candia, jhullya.e.s.martins, daniela.catelan}@ufms.br

Abstract. *This paper addresses the impact of Approximate Computing on the 64-bit RISC-V architecture, aiming for a balance between performance and energy consumption. It proposes the use of mixed instructions (such as `m_addx`, `m_subx`, and `m_mulx`), which process the 32 least significant bits approximately and the 32 most significant bits exactly. The evaluation was conducted using the GEM5 simulator, comparing them with both exact and fully approximate versions. The results demonstrate that the mixed approach can be used as an alternative, maintaining precision close to that of the exact instruction despite modest reductions in energy consumption.*

Resumo. *Este artigo aborda o impacto da Computação Aproximada sob a arquitetura RISC-V 64 bits, visando o equilíbrio entre desempenho e consumo de energia. Ele propõe o uso de instruções mistas (como `m_addx`, `m_subx` e `m_mulx`), as quais processam os 32 bits menos significativos de forma aproximada e os 32 bits mais significativos de forma exata. A avaliação foi realizada através do simulador GEM5, comparando-as com a versão exata e com a totalmente aproximada. Os resultados mostram que a abordagem mista pode ser usada como uma alternativa, mantendo a precisão próxima ao da instrução exata, apesar das reduções modestas no consumo de energia.*

1. Introdução

Com a crescente demanda por desempenho computacional, associada às restrições de consumo energético e área de hardware, a Computação Aproximada (CA) [Leon 2023] tem se consolidado como uma alternativa para aplicações tolerantes a erros. Nesse contexto, a aproximação pode ser explorada em diferentes níveis do sistema, incluindo o nível de instrução, permitindo incorporar operações aritméticas aproximadas diretamente à arquitetura do processador.

No contexto da arquitetura RISC-V [RISC-V 2022], o trabalho de [Catelan 2025] propôs instruções aritméticas totalmente aproximadas demonstrando o potencial da abordagem na obtenção de ganhos de eficiência. Entretanto, a aproximação de todos os bits pode resultar em erros elevados, limitando sua aplicação em cenários que exigem maior controle da precisão. Como alternativa, este trabalho propõe as instruções mistas `m_addx`, `m_subx` e `m_mulx`, nas quais os 32 bits mais significativos são processados de forma exata e os 32 bits menos significativos de forma aproximada.

As instruções propostas são integradas a uma versão modificada do RISC-V e avaliadas por meio do simulador GEM5 [Binkert 2011]. A avaliação considera métricas de precisão, desempenho e eficiência energética, comparando as instruções mistas com suas correspondentes exatas e totalmente aproximadas, a fim de investigar o compromisso entre redução de custos computacionais e preservação da precisão.

2. Metodologia

As instruções mistas foram desenvolvidas adotando uma abordagem híbrida, na qual as operações de soma, subtração e multiplicação são realizadas de forma aproximada nos bits menos significativos e de forma exata nos bits mais significativos.

O trecho de Código 1 apresenta a implementação da instrução mista `m.addx`. Pode-se notar que os bits menos significativos utilizam a soma aproximada (linha 3) e os mais significativos a soma exata (linha 7).

```

1 for (int i = 0; i < 32; i++) {
2     ...
3     RegVal si = (xi ^ yi);
4     result |= si;}
5 for (int i = 32; i < 64; i++) {
6     ...
7     RegVal cout_raw = ((xi & yi) | (xi & carry) | (yi & carry));
8     RegVal si = (xi ^ yi ^ carry);
9     result |= si;}

```

Código 1. Trecho de código da instrução `m.addx`.

O trecho de Código 2 apresenta a implementação da instrução mista `m.subx`, que segue o mesmo padrão da `m.addx`. Nota-se que os bits menos significativos utilizam a subtração aproximada (linha 3 e 4) e os mais significativos a subtração exata (linha 6 e 7).

```

1 //===== Aproximado (dentro do primeiro "for loop")
2     RegVal cout_raw = (((xi ^ ~yi) & borrow) | (~xi & yi));
3     RegVal si = ( (~xi & yi) | (yi & borrow)
4                 | (~xi & borrow) | (xi & ~yi & ~borrow) );
5 //===== Exato (dentro do segundo "for loop")
6     RegVal cout_raw = ((~xi & yi) | (~xi & borrow) | (yi & borrow));
7     RegVal si = (xi ^ yi ^ borrow);

```

Código 2. Trecho de código da instrução `m.subx`.

Como na multiplicação convencional, somam-se os operandos para compor o produto. Assim, a multiplicação de 64 bits resulta em um produto de 128 bits: 64 são somados de maneira aproximada (Código 3, linha 2) e os demais de forma exata.

```

1 //===== Aproximado (dentro do primeiro "for loop")
2     rk ^= (ai & bj);
3 //===== Exato (dentro do segundo "for loop")
4     rk += (ai & bj);

```

Código 3. Trecho de código da instrução `m.mulx`.

3. Resultados e Discussões

Os resultados apresentados na Tabela 1 mostram que as implementações aproximadas apresentam pequenas reduções em relação à exata para a aplicação Huffman [Huffman 1952], com destaque para a configuração 2 `mulx`; 1 `addx`. Apesar de

poucos ganhos, a aproximação mista também preserva o comportamento de desempenho e energia da implementação exata, constituindo uma alternativa intermediária. Embora as instruções totalmente aproximadas economizem mais energia, a aproximação total pode gerar erros elevados que inviabilizam certas aplicações. Assim, abdicar de um ganho energético expressivo em troca de um maior controle de precisão é vantajoso em cenários que exigem integridade nos bits mais significativos, mas que ainda se beneficiam de uma redução segura nos bits menos significativos.

Tabela 1. Resultados das implementações exata, aproximadas e mistas.

Implementação	Tempo (μs)	Energia (μJ)	Potência (μW)
Original	146,360	748,334	5,113
<i>Totalmente aproximadas</i>			
1 mulx; 0 addx	146,310	747,901	5,112
1 mulx; 1 addx	146,240	747,491	5,111
2 mulx; 1 addx	146,270	747,410	5,110
<i>Mistas</i>			
1_m_mulx; 0_m_addx	146,340	748,309	5,113
1_m_mulx; 1_m_addx	146,270	748,060	5,114
2_m_mulx; 1_m_addx	146,300	748,237	5,114

4. Conclusão

Este estudo demonstrou que as instruções mistas representam uma alternativa viável para aplicações tolerantes a erros na arquitetura RISC-V 64 bits. Embora as reduções no consumo de energia tenham sido modestas, cerca de 0,12% a 0,35% em relação à exata, as instruções mistas mantiveram o tempo de execução praticamente inalterado, com variações inferiores a 0,1%. A configuração 1_m_mulx; 1_m_addx alcançou 748,060 μJ e 146,270 μs , valores próximos aos da execução exata (748,334 μJ e 146,360 μs), enquanto a totalmente aproximada obteve 747,410 μJ . As instruções mistas oferecem maior controle sobre a precisão por preservarem a exatidão nos 32 bits mais significativos, constituindo-se uma solução intermediária promissora entre as abordagens exata e totalmente aproximada, especialmente em cenários com restrições energéticas e onde erros em bits menos significativos são toleráveis. Apesar dos resultados obtidos demonstrarem que a abordagem mista atua como um equilíbrio promissor entre exatidão e economia de energia, a análise de um único cenário limita a generalização das conclusões. Portanto, propõe-se como trabalho futuro a avaliação das instruções propostas em aplicações de outros domínios, tais como processamento digital de sinais (DSP) e filtros de imagem.

Referências

- Binkert, N. e. a. (2011). The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7.
- Catelan, D. L. (2025). Approximate computing: Contributions to the design of arithmetic circuits and instruction-set architectures. *Repositório Institucional da UFMS*.
- Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.
- Leon, V. e. a. (2023). Approximate computing survey, part i: Terminology and software e hardware approximation techniques.
- RISC-V (2022). Risc-v toolchain. Available at: <https://github.com/riscv-collab/riscv-gnu-toolchain>. Accessed: June/2026.