

ANÁLISE DA ARQUITETURA DE HARDWARE DO NES

Rafaela Dias Mori Cruz ⁽¹⁾, Claus Cornelio dos Santos Filho ⁽²⁾, Giovanna Oliveira de Jesus ⁽³⁾, Guilherme Gomes Ribeiro ⁽⁴⁾, Guilhermy Santana Bernardo da Silva ⁽⁵⁾, Julia Martins Pereira ⁽⁶⁾, Nathan Ferreira Neves ⁽⁷⁾. Orientador: Prof. Dr. Fernando Trevisan Saez Parra. ⁽¹⁾6-ENGCMP-00349802, ⁽²⁾6-ENGCMP-00350915, ⁽³⁾6-ENGCMP-00349647, ⁽⁴⁾6-ENGELE-00349318, ⁽⁵⁾6-ENGELE-00360326, ⁽⁶⁾6-ENGCMP-00349371, ⁽⁷⁾6-ENGCMP-00350327.

RESUMO

O projeto analisa a arquitetura do console Nintendo Entertainment System (NES) por meio do desenvolvimento de um jogo autoral em Assembly 6502 [1,2]. O estudo, realizado em parceria com a empresa J Falsarella – NE, tem como objetivo compreender como os recursos limitados do hardware influenciam o design e o desempenho dos sistemas de 8 bits [3,4]. Para isso, utiliza-se o processador Ricoh 2A03 e a interação entre a CPU e a PPU (*Picture Processing Unit*), elementos fundamentais para a geração de gráficos e controle da execução do jogo [1,5,6]. O código foi desenvolvido, compilado e testado em ambiente de emulação, permitindo observar na prática os desafios de otimização e temporização característicos da programação de baixo nível [2,7,9]. Os resultados parciais demonstram que a eficiência e a criatividade são determinantes para superar as restrições impostas pelo hardware [10,11]. O estudo reforça a importância de compreender a base dos sistemas computacionais clássicos para aprimorar o raciocínio técnico e desenvolver soluções mais otimizadas em projetos modernos [12,15].

Palavras-chave: NES; Assembly 6502; Ricoh 2A03; arquitetura de hardware; programação de baixo nível.

1.Introdução

Hoje em dia, criar jogos ficou muito mais acessível graças às engines modernas que permitem até mesmo quem não tem formação técnica desenvolver jogos 2D ou 3D. Mas, apesar de toda essa facilidade, acredita-se que entender como funcionavam os sistemas mais antigos — aqueles que realmente moldaram a indústria — ainda é fundamental para quem quer dominar os conceitos reais por trás da arquitetura de computadores.

Lançado nos EUA em 1985 (como versão internacional do Famicom japonês, de 1983), o NES chegou num momento delicado: o mercado de videogames nos Estados Unidos estava em crise, com o público desacreditado e várias empresas falindo. Ainda assim, a Nintendo conseguiu se destacar e revitalizar o setor, graças a uma combinação de estratégia de marketing eficiente e um hardware inovador para a época, baseado no processador MOS 6502. Programar para essa máquina não era tarefa fácil: era preciso escrever diretamente em Assembly, uma linguagem de baixo nível que exige um conhecimento profundo do hardware e de cada instrução da CPU.

Por isso, este trabalho analisa a arquitetura do NES, desenvolvendo um jogo próprio em Assembly 6502. O projeto está sendo realizado em parceria com a empresa **J Falsarella -NE**, que atua como cliente neste estudo e apoia a aplicação prática dos conceitos explorados. O jogo, na verdade, serve como uma espécie de “laboratório”: ao tentar fazer algo funcionar dentro das limitações reais do console — como memória reduzida, controle da PPU (*Picture Processing Unit*) e uso de mappers —, entender na prática como o sistema realmente opera. Ao longo do semestre, será aprofundado esses componentes e, no final, entregar não só um jogo funcional, mas também uma reflexão técnica sobre como a arquitetura do NES influencia (e limita) o que é possível criar nele — tudo isso mantendo o espírito de entretenimento que fez essa plataforma se tornar lendária.

2 . Metodologia

Este trabalho adota uma abordagem prática e experimental para a análise da arquitetura do Nintendo Entertainment System (NES), com base no desenvolvimento de um programa funcional em linguagem Assembly 6502. A metodologia empregada combina o uso de ferramentas de desenvolvimento acessíveis, documentação técnica especializada e testes em ambiente de emulação, permitindo uma compreensão direta do funcionamento interno do console.

O processo inicia-se com a escrita do código-fonte no editor Visual Studio Code (VS Code), escolhido por sua leveza e suporte à sintaxe personalizada. Nesse ambiente, o programa é desenvolvido em Assembly 6502, respeitando as particularidades da arquitetura do NES, como a organização das seções de código, dados e vetores de interrupção. Após a redação ou modificação do código, o arquivo é salvo com o nome `coresandam2.asm` na pasta contendo o compilador NESASM — um assembler específico para o NES, extraído do pacote `nesasm.zip` previamente baixado. Esse compilador é responsável por converter o código Assembly em uma ROM no formato `.nes`, compatível com emuladores e com o hardware original do console.

Para gerar a ROM, utiliza-se o comando contido no arquivo `compilar.txt`: `nesasm.exe coresandam3.asm` (pode ser qualquer nome). Esse comando é copiado e executado no Prompt de Comando dentro da mesma pasta do NESASM. A execução gera automaticamente o arquivo `coresandam2.nes`, que corresponde à ROM pronta para testes.

Em seguida, a ROM é carregada no emulador FCEUX, ferramenta amplamente utilizada pela comunidade de desenvolvedores de homebrew para o NES. Além de executar o programa, o FCEUX oferece recursos avançados de depuração, como visualização da memória, inspeção dos registradores da CPU e monitoramento do estado da PPU (Picture Processing Unit), permitindo uma análise detalhada do comportamento do código em tempo de execução.

O programa desenvolvido exibe quatro quadrados coloridos na tela (azul, amarelo, roxo, vermelho e verde). Ao pressionar as teclas de direção do teclado — que simulam o D-pad do controle do NES —, as cores dos quadrados são alteradas. Embora aparentemente simples, essa funcionalidade exige manipulação direta da PPU, uso correto das tabelas de tiles, definição de paletas de cores e configuração da tabela de atributos, tudo implementado manualmente em Assembly 6502, sem abstrações de alto nível.

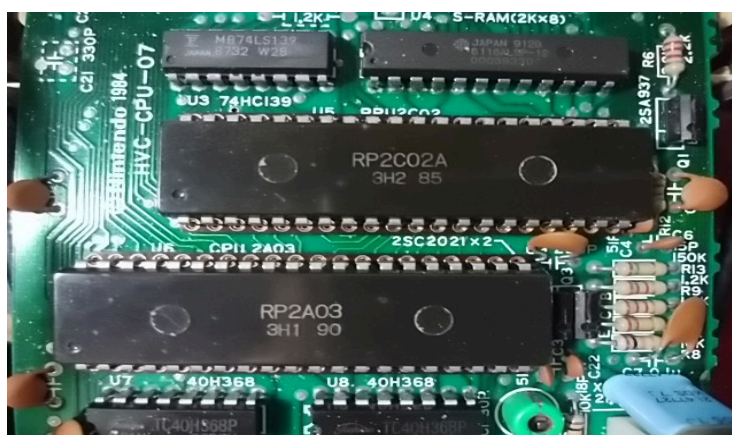
Esse ciclo — escrita do código no VS Code, compilação com NESASM e teste no FCEUX — é repetido a cada iteração do desenvolvimento, permitindo ajustes rápidos e uma compreensão progressiva das limitações e características do hardware do NES, como os 2 KB de RAM disponíveis, a frequência de clock de 1,79 MHz e o conjunto restrito de instruções do processador Ricoh 2A03 (baseado na arquitetura MOS 6502).

Além das ferramentas práticas, a pesquisa também se apoia em documentação técnica especializada, incluindo datasheets do chip Ricoh 2A03 e o NesDev Wiki, referência consolidada na comunidade de desenvolvedores. Esses materiais auxiliam na validação das decisões de implementação e no aprofundamento teórico sobre o funcionamento do sistema.

Dessa forma, a metodologia empregada integra experimentação prática, análise técnica e referencial teórico, com o objetivo claro de investigar a arquitetura do NES por meio da programação em Assembly 6502.

Figura 1 - Processador que está sendo emulado é o Ricoh 2A03.

[Fonte: love80s.ru]



3. Desenvolvimento

O trabalho está sendo construído na prática: o código foi escrito em Assembly 6502 no VS Code, o mesmo foi transformado em uma ROM do NES usando o NESASM e, em seguida, testado tudo no emulador FCEUX. Parece simples — e é —, mas é justamente essa simplicidade que permite uma conexão direta com o hardware. Cada linha escrita vira algo visível na tela, e isso faz toda a diferença na hora de entender como o NES realmente funciona.

A tela inteira é montada a partir de pequenos blocos chamados tiles — quadrados de 8×8 pixels que funcionam como peças de um quebra-cabeça. Para que os quadrados apareçam na posição certa, esses tiles precisam ser colocados exatamente nos endereços corretos da memória de vídeo. As cores não são livres: o NES trabalha com paletas limitadas, e para definir qual cor vai onde, usamos uma estrutura chamada tabela de atributos. Parece complicado no papel, mas só faz sentido mesmo quando você vê (e corrige!) o resultado na tela.

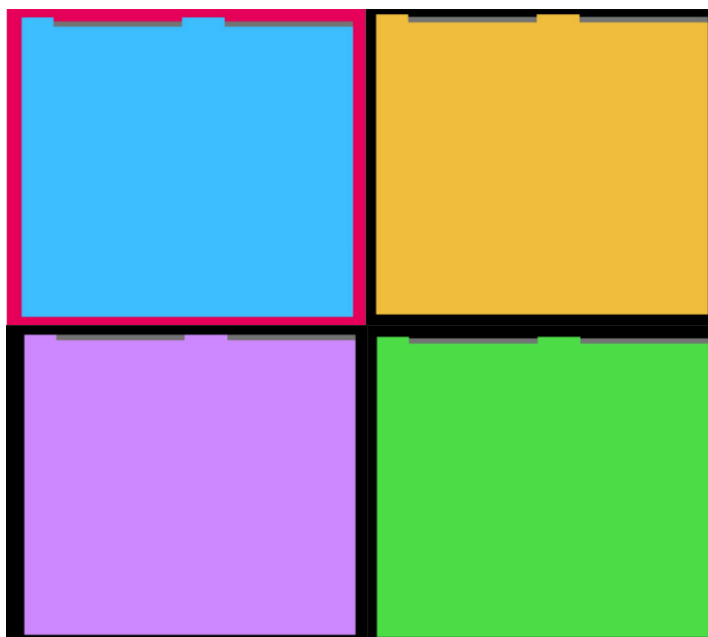
Esse tipo de experiência mostra que programar para o NES não é só sobre lógica — é sobre entender a máquina. E é essa conversa direta entre código e hardware que torna o aprendizado tão concreto.

Além disso, tudo tem que acontecer na hora certa. A PPU (a parte do NES que gera a imagem) só aceita mudanças na memória durante um intervalo chamado vblank — um breve momento em que o sinal de vídeo “dá uma pausa” entre um quadro e outro. Se escrever fora desse período, a imagem treme, some ou vira um borrão. Por isso, o programa inclui uma rotina que espera justamente por esse instante antes de atualizar qualquer coisa na tela. Era assim que os jogos antigos funcionavam.

Depois de pronto, o código vira uma ROM (um arquivo .nes) com um único comando no NESASM. Em segundos, essa ROM está pronta para rodar no FCEUX. Lá, além de ver os quadrados, dá para interagir: ao apertar as setas do teclado (que simulam o controle do NES), as cores mudam. Isso envolve ler os botões pressionados e atualizar a paleta em tempo real — sempre respeitando, de novo, o ciclo do vblank.

A parte de escrever, compilar, testar e ajustar tem sido essencial para entender como os desenvolvedores da época faziam tanto com tão pouco: 2 KB de memória, um processador lento e zero recursos extras. Estes testes permitiram validar o funcionamento do jogo proposto e compreender na prática as limitações do hardware do NES e o que os tornava especiais era o domínio absoluto sobre cada detalhe.

Figura 2 - Exemplo de execução do emulador FCEUX 2.6.6, ferramenta utilizada para análise e programação de cores e debug de jogos do NES.
[Fonte: Próprio Autor/Arquivo Pessoal]



4. Considerações Finais

O estudo da arquitetura do NES e da linguagem Assembly 6502 mostra como a criatividade e o domínio técnico foram essenciais para superar as limitações da época mesmo com poucos recursos, desenvolvedores conseguiram criar jogos que marcaram gerações, provando que inovação não depende apenas de poder de processamento, mas de engenhosidade e dedicação.

Mais do que entender o funcionamento de um console antigo, esse estudo mostra o quanto conhecer as bases da computação faz diferença. Trabalhar com o hardware e lidar com o sistema de 8 bits traz uma visão mais clara de como os jogos funcionam, algo que muitas vezes se perde com as facilidades das ferramentas modernas.

Revisitar essa tecnologia também é uma forma de valorizar o que muitas vezes passa despercebido, o esforço, a criatividade e a paixão de quem trabalha com recursos limitados. O NES se tornou especial não apenas pela tecnologia que trouxe na época, mas principalmente pelo papel fundamental que desempenhou na história dos videogames, abrindo caminho para uma das maiores formas de entretenimento.

5. Referências

- [1] NES documentation [Internet]. NESDev Archive; c2004 [acesso em 2025 set 10]. Disponível em: <https://archive.nesdev.org/NESDoc.pdf>
- [2] Darlison J. An introduction to 6502 assembly and low level programming [Internet]. Codeburst; 2018 [acesso em 2025 set 12]. Disponível em: <https://codeburst.io/an-introduction-to-6502-assembly-and-low-level-programming-7c11fa6b9cb9>
- [3] Reddit. I tried to learn assembly (specifically 6502) and... [Internet]. Reddit; 2023 [acesso em 2025 set 15]. Disponível em: https://www.reddit.com/r/asm/comments/17vkt1q/i_tried_to_learn_assembly_specifically_6502_and/?t=pt-br
- [4] NES Architecture – A Practical Analysis [Internet]. [s.l.]: [s.n.]; [acesso em 2025 set 18].
- [5] Battleline Games. NES Starter Kit [Internet]. GitHub; 2022 [acesso em 2025 set 20]. Disponível em: <https://github.com/battlelinegames/nes-starter-kit>
- [6] Kopp H. Nintendo Entertainment System programming setup [Internet]. 2020 [acesso em 2025 set 23]. Disponível em: <https://hkopp.github.io/2020/03/nintendo-entertainment-system-programming-setup>
- [7] Patater. NESASM Assembler for the NES [Internet]. Patater.com; 2003 [acesso em 2025 set 26]. Disponível em: <https://www.patater.com/gbaguy/nesasm.htm>
- [8] Visual Basic Forums. NES 6502 Programming Tutorial – Part 1: Getting Started [Internet]. VBForums; 2019 [acesso em 2025 set 29]. Disponível em: <https://www.vbforums.com/showthread.php?858389-NES-6502-Programming-Tutorial-Part-1-Getting-Started>
- [9] Hughson M. How to start making NES games [Internet]. 2021 [acesso em 2025 out 2]. Disponível em: <https://www.matthughson.com/2021/11/17/how-to-start-making-nes-games/>
- [10] Leventhal L. How to program the 6502 [Internet]. Google Books; [acesso em 2025 out 5]. Disponível em: <https://books.google.com.br/books?id=oxuvDwAAQBAJ>
- [11] Zaks R. Programming the 6502 [Internet]. Google Books; [acesso em 2025 out 8]. Disponível em: <https://books.google.com.br/books?id=cwcOEQAQBAJ>

[12] Universidade de Alicante. Programming in Assembly Language [Internet]. RUA Repositório Institucional; [acesso em 2025 out 12]. Disponível em: <https://rua.ua.es/entities/publication/e6063e1f-28ec-4a44-bf98-56d2d5cbd670>

[13] Silva JL. Programação Assembly e C para PIC [Internet]. 2017 [acesso em 2025 out 16]. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/52209894/03_ASM_C-libre.pdf

[14] Terra. Por que o NES se chamava Famicom no Japão? [Internet]. Terra GameOn; 2023 [acesso em 2025 out 21]. Disponível em: <https://www.terra.com.br/gameon/plataformas-e-consoles/por-que-o-nes-se-chamava-famicom-no-japao,53f63e434e6b08f1006236323ebd0183fj10h19g.html>

[15] 6502 Assembly Language Programming [Internet]. [s.l.]: [s.n.]; [acesso em 2025 out 25].