RARSCOV24

Antonio B. Coelho, Leonardo M. M. K. Guimarães, Rafael M. Alencar

Departamento de Ciências da Computação
Universidade de Brasília (UnB) – Brasília, DF – Brazil

Abstract. *This work was developed for the course Introduction to Computer Systems (ISC), taught by Professor Dr. Marcus Vinícius Lamar. The game is an adaption of the classic "Pac-Man", from 1980, programmed in Assembly for the RISC-V Architecture, using the RARS Simulator.*

Keywords— *Assembly, RISC-V, RARS, Pac-Man*

Resumo. *Este é um trabalho desenvolvido para a matéria de Introdução aos Sistemas Computacionais (ISC), ministrada pelo professor Dr. Marcus Vinícius Lamar. O jogo é uma adaptação do clássico "Pac-Man", de 1980, desenvolvido em Assembly para a arquitetura RISC-V, por meio do Simulador RARS.*

Palavras-chave— *Assembly, RISC-V, RARS, Pac-Man*

1. Introdução

Desde seu início, o desenvolvimento de jogos digitais passou por inúmeras revoluções e inovações que culminaram no cenário atual, dominado por *game engines* e pela programação orientada a objeto. No entanto, algumas décadas atrás, quando as limitações do hardware e dos sistemas computacionais ainda eram consideráveis, o desenvolvimento de games era baseado na simplicidade e na eficiência do código estruturado. Foi nesse contexto tecnológico que surgiu o icônico "Pac-Man", lançado pela empresa de jogos *Namco* (atual *Bandai Namco*) no fim de maio de 1980.

Figura 1: Logotipo do jogo original

O conceito do jogo é simples: Pac-Man deve consumir todas as bolas (ou pastilhas) no labirinto enquanto evita ser capturado pelos fantasmas que o perseguem. Ao comer uma pastilha especial, ele ganha o poder temporário de devorar os fantasmas. Embora o objetivo pareça direto, o sistema de pontuação, os recursos de *power-up* e os diferentes padrões de movimento de cada fantasma acrescentam uma camada de desafio e competitividade para o jogador e tornam a recriação de *Pac-Man* uma tarefa muito mais ambiciosa do que aparenta à primeira vista.

2. Metodologia

O trabalho foi elaborado utilizando o Simulador RARS para a ISA (“Instruction Set Architecture”) RV32-I de processadores com arquitetura RISC-V.

2.1 Arquitetura RISC-V

RISC-V é uma Arquitetura de Conjunto de Instruções (ISA) de código aberto, baseada no princípio de Computação com Conjunto Reduzido de Instruções (RISC).

Desenvolvida em 2010, na Universidade da Califórnia, em Berkeley, ela é projetada para ser modular, simples e extensível, o que permite seu uso em uma ampla gama de dispositivos (desde sistemas embarcados até computadores de alto desempenho).

Seu caráter de código aberto a torna altamente acessível, permitindo que qualquer pessoa a implemente e personalize sem taxas de licenciamento.

A simplicidade e a escalabilidade do RISC-V tornam-no ideal para o desenvolvimento de hardware personalizado, e sua crescente adoção na indústria de tecnologia destaca sua flexibilidade e sua eficiência.

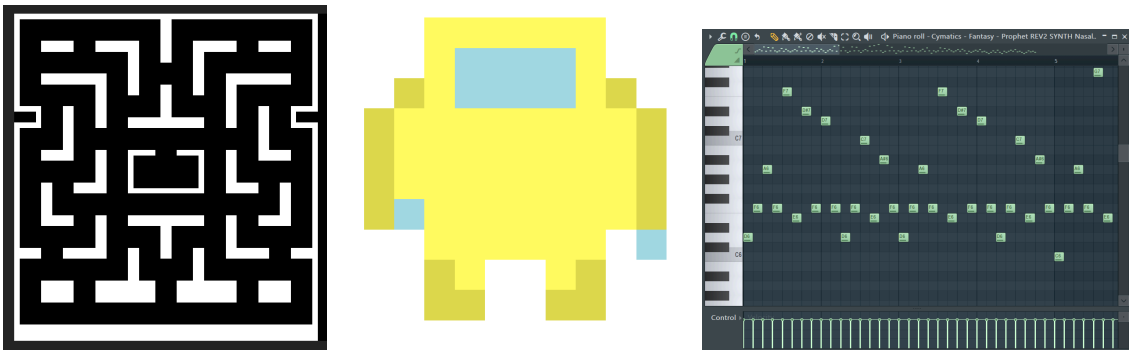
2.2 Simulador RARS

RARS (RISC-V Assembler and Runtime Simulator) é um simulador de código aberto projetado para rodar programas escritos na linguagem Assembly para a arquitetura RISC-V. Ele permite a montagem, simulação e depuração de programas, oferecendo uma interface gráfica intuitiva para acompanhar a execução do código em tempo real.

Criado para fins educacionais, o RARS é amplamente utilizado em cursos de arquitetura de computadores e sistemas operacionais, proporcionando um ambiente fácil de usar para o aprendizado de Assembly e a compreensão da arquitetura RISC-V.

2.3 Pré-desenvolvimento

Inicialmente, foram elaborados os gráficos dos mapas (mapas das fases 1 e 2 e os mapas de remédios), dos modelos do médico, dos vírus, dos remédios e as músicas de cada fase.



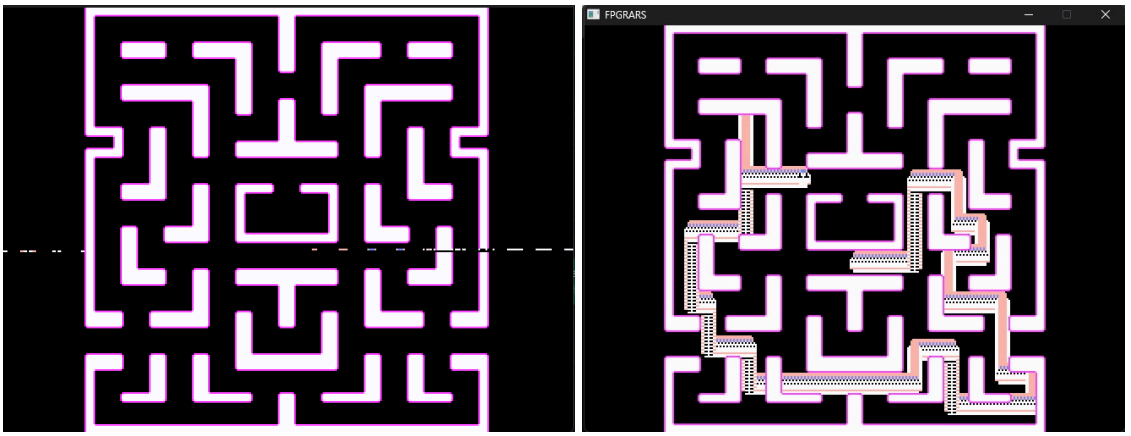
Figuras 2, 3 e 4: Mapa, personagem e música

Em sequência, iniciou-se a esquematização do que seria, posteriormente, o código do jogo. Foram separados em partes cada um dos mecanismos que foram implementados para que o jogo funcionasse de maneira semelhante à original.

2.4 Desenvolvimento em Assembly

Após a definição das etapas a serem realizadas, foram programadas as primeiras funcionalidades: exibição do mapa, exibição do personagem, movimentação do personagem e colisão com as paredes do mapa.

Logo no início, foram encontradas algumas dificuldades, principalmente relacionadas à falta de familiaridade com a programação em baixo nível. A construção de laços e os saltos condicionais rapidamente se tornaram naturais. Entretanto, lidar com o *Bitmap Display* continuou sendo uma constante de estresse e frustração até o fim do projeto.



Figuras 5 e 6: Tentativas de mostrar o personagem na tela

Posteriormente, foram implementados música, efeitos sonoros, remédios, contagem de pontuação e condição de vitória. Ao longo do desenvolvimento, também foram alterados alguns detalhes gráficos, de modo a facilitar a jogabilidade e tornar o jogo mais bonito. Vale destacar que a colaboração entre os grupos foi essencial durante todo o processo, e muito do que foi feito não teria sido possível sem essa ajuda mútua.

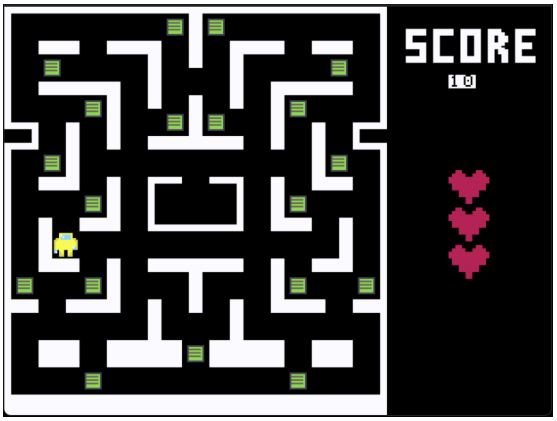


Figura 7: Jogo com pontuação e HUD implementados

3. Resultados Obtidos

O jogo conta com a maioria das funcionalidades essenciais planejadas, incluindo a exibição dos mapas, músicas, personagens (animados) e jogabilidade funcional. A contagem de pontos acontece à medida que o médico captura as moedas e há a mudança de fase quando o jogador vence (há, também, um "cheat" que muda de fase, para fins de apresentação).

Entretanto, nem todas as mecânicas inicialmente previstas foram concluídas devido à complexidade do projeto e às habilidades limitadas de organização de tempo dos integrantes do grupo. Entre os elementos que ficaram pendentes para desenvolvimento futuro, destaca-se a implementação dos fantasmas, parte fundamental do "Pac-Man", que requer um nível adicional de refinamento no código.

4. Conclusão

Desenvolver, ainda que parcialmente, um jogo em Assembly foi uma experiência rica em aprendizado e proporcionou ao grupo um entendimento mais profundo do funcionamento dos sistemas computacionais.

A experiência de programar em Assembly aproxima muito o desenvolvedor da essência do processamento binário, permitindo um contato quase que direto com a "linguagem de zeros e uns". Esse nível de proximidade com a arquitetura do computador gera um impacto positivo na formação dos estudantes de Ciência da Computação, pois, apesar da complexidade envolvida, é uma atividade crucial para entender por completo o funcionamento de sistemas computacionais além de trabalhar habilidades desejáveis para quem deseja atuar na área de sistemas de baixo nível.

O desenvolvimento de jogos, em particular, exige uma lógica criativa, desafiando os alunos a saírem de suas zonas de conforto. Isso os leva a repensar estruturas, otimizar recursos e considerar aspectos de desempenho que, muitas vezes, passam despercebidos em linguagens de alto nível. Esses desafios práticos não apenas reforçam o aprendizado de Assembly, mas também desenvolvem a capacidade de resolver problemas de maneira criativa e eficiente.

Referências Bibliográficas

França Lisboa, Victor H. "LAMAR - Learning Assembly for Machine Architecture and RISC-V", https://github.com/victorlisboa/LAMAR

Neves de Macedo, Nirva "Documentação e referência para o RARS versão UnB", https://nirvacx.github.io/RarsHelp/

RISC-V International - "History of RISC-V", https://riscv.org/about/history/

Birch, Chad "GameInternals - Understanding Pac-Man ghost behavior" https://gameinternals.com/understanding-pac-man-ghost-behavior