Relatório Projeto Final
Grupo 5

Hugo Macedo Ramos * Lucas Teixeira de Souza † Marcelo Vitor Meira de Lucena ‡ Tiago Leão Buson §

Universidade de Brasília, 17 de setembro de 2024

RESUMO

Este relatório descreve o desenvolvimento de um jogo baseado no clássico Metroid (NES) utilizando Assembly RISC-V (ISA RV32IMF), como parte do projeto final da disciplina de Organização e Arquitetura de Computadores (OAC). O jogo foi implementado para ser executado em uma FPGA, com suporte a monitor, teclado e caixa de som. O objetivo principal do projeto foi aplicar conceitos de programação em baixo nível e organização de processadores, abordando movimentação, combate, inteligência artificial de inimigos, itens e design de níveis.

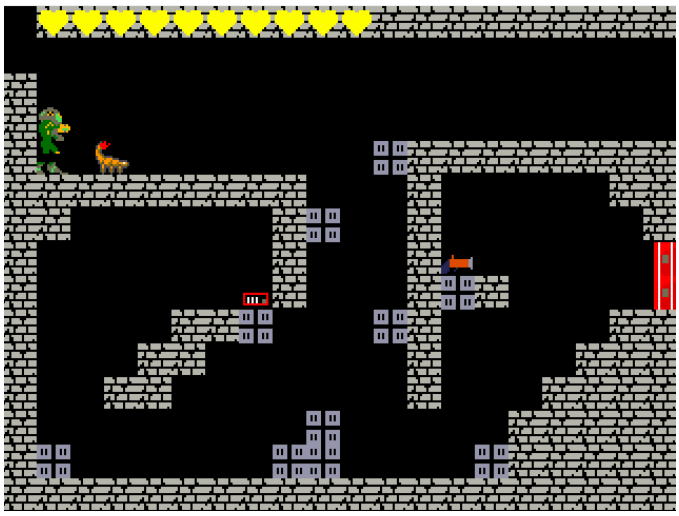

Figura 1: *Print jogo*

Figura 2: *Metroid*

Palavras-chave: OAC · Assembly RISC-V · Metroid · FPGA · ISA RV-32IMF · Gameloop · GitHub ·

1 INTRODUÇÃO

O projeto final da disciplina de Organização e Arquitetura de Computadores (OAC) consiste na criação de um jogo temático utilizando a linguagem Assembly RISC-V, especificamente na ISA RV32IMF, com foco na aplicação prática dos conhecimentos adquiridos sobre organização de processadores e programação em baixo nível. O tema escolhido para o desenvolvimento do jogo foi Metroid, clássico do console NES, e a implementação foi feita em uma FPGA, utilizando periféricos como monitor e teclado para interação. Este relatório detalha o processo de desenvolvimento do jogo, os desafios enfrentados e as soluções adotadas, ressaltando o uso da arquitetura RISC-V para criar uma experiência interativa fiel ao jogo original.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

O projeto é baseado no jogo Metroid, utilizamos alguns vídeos de gameplays do jogo original, exemplos de implementação de antigos monitores e conversores de arquivos disponíveis no GitHub. Além disso, os principais trabalhos relacionados utilizados foram os projetos de alunos dos semestres anteriores disponíveis na plataforma do Aprender 3.

3 METODOLOGIA

O desenvolvimento do jogo foi realizado inteiramente em Assembly RISC-V, utilizando a ISA RV32IMF, com foco em implementar as principais mecânicas do Metroid original, como movimentação, ataques e interação com inimigos e itens. O código foi projetado para ser executado em uma FPGA conectada a periféricos como monitor, teclado e caixa de som, com a integração de diferentes componentes do hardware.

*221037634@aluno.unb.br
†221017041@aluno.unb.br
‡221030034@aluno.unb.br
§200034162@aluno.unb.br

O desenvolvimento do jogo seguiu uma abordagem estruturada de renderização de frames, similar a um sistema de turnos, em que cada entidade no jogo – Samus e inimigos – possui um "turno"para realizar suas ações durante cada frame de execução. Essa abordagem permitiu a implementação de mecânicas de movimentação, combate e interação com o cenário de forma organizada e eficiente.

A lógica do jogo foi baseada no ciclo de um game loop, que é responsável por renderizar um novo frame a cada iteração. Durante o ciclo de cada frame, todas as entidades têm a oportunidade de realizar suas ações, conforme sua lógica programada.

3.1 Divisão

A implementação seguiu uma estrutura modular, dividida nas seguintes etapas:

3.2 Renderização de Frame e Sistema de Turnos

O jogo foi implementado de maneira que cada entidade (Samus e os inimigos) possua um turno dentro de cada frame. A cada ciclo do game loop, o código decide as ações que cada personagem deve tomar com base em seu estado atual. Primeiramente, a Samus recebe seu turno, no qual é avaliado o input do jogador (movimentação, ataque, pulo, etc.). Após isso, os inimigos são processados em sequência, cada um com suas respectivas lógicas de IA.

A renderização do frame ocorre após a resolução dos turnos de todas as entidades. Durante o turno de cada personagem, sua posição e animação são atualizadas conforme a ação realizada (andar, atacar, pular, etc.), e, uma vez que todos os turnos são concluídos, o estado final de cada entidade é desenhado na tela. Esse sistema garante que todas as ações estejam sincronizadas e processadas antes de o próximo frame ser exibido.

3.3 Movimentação e animação do personagem jogável

A movimentação da Samus foi implementada com base em um sistema de coordenadas que controla o movimento horizontal e vertical, além de animações associadas a diferentes estados, como pular, agachar e atacar. A inércia foi considerada ao calcular os saltos e a direção da Samus, e os frames de animação foram trocados dinamicamente conforme a interação do jogador.

3.4 Ataques e interação com inimigos

A Samus possui a capacidade de atirar projéteis que causam dano aos inimigos. Cada tipo de inimigo, incluindo um chefe, foi programado com uma inteligência artificial específica para reagir aos movimentos da Samus e atacar o jogador de maneiras distintas. A IA dos inimigos utiliza um sistema de detecção de colisão para determinar quando atacar ou se mover.

3.5 Itens e acessibilidade a novas áreas

Foram implementados dois itens principais no jogo. Um deles permite à Samus acessar novas áreas, enquanto o outro melhora a capacidade de combate da personagem. Esses itens são obtidos ao explorar as salas e derrotar inimigos, e são armazenados e exibidos junto com as informações de vida e equipamento da Samus.

3.6 Design de níveis e salas

O jogo contém três salas distintas, separadas por portas. O jogador deve obter chaves para abrir as portas e progredir.

Por limitações de tempo e recursos, não foi possível implementar o sistema de música e efeitos sonoros. Contudo, todas as outras funcionalidades foram concluídas conforme os requisitos estabelecidos para o projeto.

4 Resultados Obtidos

Com isso, conseguimos implementar uma grande parte das funcionalidades solicitadas. É possível movimentar o personagem, atacar os inimigos e ser atacado por eles, acessar novas áreas do mapa conforme a coleta dos itens e também a renderização das diferentes salas do mapa.

5 Conclusões e Trabalhos futuros

O desenvolvimento deste projeto em Assembly RISC-V permitiu uma aplicação prática dos conceitos aprendidos sobre organização de processadores e programação em baixo nível, especialmente no que diz respeito à integração de hardware e software em uma FPGA. A recriação do jogo Metroid demonstrou a capacidade de lidar com movimentação, inteligência artificial, detecção de colisões, manipulação de itens e ambientes interativos, além de apresentar desafios relacionados à otimização do código e ao gerenciamento de recursos limitados.

Esse projeto proporcionou um entendimento mais profundo dos desafios envolvidos na programação de baixo nível, como controle direto de hardware e otimização de desempenho, ao mesmo tempo em que consolidou habilidades essenciais para o desenvolvimento de software voltado para sistemas embarcados e arquiteturas específicas.

6 Referências Bibliográficas

Gameplay Metroid
Sprites
Tutorial de RISC-V
Livrão de OAC
ChatGPT