Relatório Trabalho Final - Grupo G7
Pokemon Assembly Version

Daniel da Cunha Pereira Luz [*] Gustavo Mello Tonnera [†] Jasiel Henrique Alves Genuino Santos [‡]
Lucas Amaral De Faria [§] Vinicius Da Silva Araujo [¶]

Universidade de Brasília, 15 de fevereiro de 2023

Figura 1: *Pokemon Assembly Version*

RESUMO

Como trabalho final da disciplina de Organização e Arquitetura de Computadores, foi proposto que os estudantes fizessem um jogo baseado no Pokémon FireRed/LeafGreen em Assembly RISC-V e o implementasse no Kit FGPA DE1-SoC. Este artigo apresenta como os integrantes do grupo G7 desenvolveram sua versão do jogo.

Palavras-chave: Oeganização e Arquitetura de Computadores · Assembly RISC-V · Pokémon

1 INTRODUÇÃO

Como forma de avaliação da disciplina de OAC, foi proposto a criação de um jogo baseado nos jogos Pokémon FireRed/LeafGreen, escrito em Assembly RISC-V (ISA RV32IMF). O trabalho tem como objetivo testar e aferir os conhecimentos de linguagem Assembly RISC-V e a organização de processadores vistos no decorrer so semestre.

Os jogos Pokémon FireRed e LeafGreen são jogos eletrônicos de RPG de 2004 recriado pelo Pokémon Red e Blue de Game Boy em 1996. Eles foram desenvolvidos pela Game Freak e publicados pela The Pokémon Company e pela Nintendo para o Game Boy Advance. FireRed e LeafGreen foram lançados pela primeira vez no Japão em janeiro de 2004 e na América do Norte e Europa em setembro e outubro de 2004, respectivamente. Os jogos fazem parte da terceira geração da série de jogos eletrônicos Pokémon e têm a distinção de serem as primeiras recriações aprimoradas de jogos anteriores da franquia.

A fim de cumprir os objetivos do trabalho, foram propostos 11 items de avaliação: 1) História do jogo; 2) Música e efeitos sonoros; 3) Mínimo de 1 item utilizável; 4) Mínimo de 5 tipos de pokémons diferentes, com um sistema de pedra-papel-tesoura" entre eles (normal perde para luta, luta para psíquico, etc.); 5) Mínimo de três tipos de telas jogáveis: seleção de pokémons iniciais, área aberta, e ginásio; 6) Fases com número crescente de inimigos (inclusa a animação deles), espaços abertos e paredes; 7) IA que controla os inimigos e sistema de turnos da luta; 8) Cenas de batalhas, podendo ser apenas uma imagem mostrando os personagens e as vidas deles; 9) Menu de ações do jogador; 10) Movimentação dos personagens; 11) Três tipos de terrenos especiais. Ao longo da sessão Resultados Obtidos, explicamos como cada um desses items foi implementado.

Figura 2: *Imagens dos jogos originais*

[*] dan08jan@gmail.com
[†] gmtonnera@gmail.com
[‡] jasiel.genuino@gmail.com
[§] lucasamaralfaria@gmail.com
[¶] vinicius.ar02@gmail.com

2 Fundamentação Teórica e Trabalhos Relacionados

O grupo fundamentou-se a parte da linguagem assembly RISC-V através dos slides[2] do material disponibilizado pelo professor durante o semestre, nesses slides são firmados os firmamentos necessários para se ter uma boa compreensão da linguagem assembly RISC-V, lá existem exemplos de como imprimir um dado na tela e de como se configura um teclado externo, esses recursos foram muito importantes, pois eles são essenciais para ter-se um jogo eletrônico. Adicionalmente foi observado um pouco do livro do Patterson[1], para alguns conceitos da linguagem de Assembly RISC-V e a organização dos processadores.

O jogo foi baseado na história original do jogo Pokémon FireRed[3], prestamos atenção em cada detalhe, como, por exemplo, a ambientação, a fala dos personagens, o layout de luta, entre outos aspectos importantes. Foram feitas algumas modificações na história integrando aspectos do ambiente do campus da Universidade de Brasília, assim como vários easter eggs.

Para muitas funcionalidades, foram utilizados conceitos de disciplinas anteriores que os integrantes estudaram na universidade, por exemplo: A criação do mapa, que foi feito por meio de uma matriz, trouxe muitos conceitos de Álgebra Linear; A criação de objetos na programação e mecanismos de algoritmo foi baseada na matéria de Estrutura de Dados.

Durante a produção do jogo, o grupo se instruiu muito pelas monitorias, onde os monitores ajudaram com dicas e resolução de dúvidas, foi ainda indicado por eles, projetos anteriores de referência que eles mesmos fizeram durante esta matéria, projetos como o OA-Celeste, Fire Emblem: The Blazing Blade em RISC-V, Castlevania: Synth of the Night em RISC-V.

3 Metodologia

Para a implementação do jogo, seguimos a documentação especificada no arquivo disponibilizado no aprender3 pelo professor. Foram utilizados os seguintes softwares para as respectivas atividades, o RARS, que é um montador RISC-V que simula o tempo de execução, para o debug durante o desenvolvimento do jogo, o Visual Studio Code, que é um editor de código-fonte desenvolvido pela Microsoft, para escrever o código, o GitHub, uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git, um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo, para a gestão de tarefas, o Gimp, um programa de código aberto voltado principalmente para criação e edição de imagens, para fazer algumas edições e o paint.net, um programa de computador gratuito e open-source utilizado na manipulação e edição de imagem e fotografia, para exportar os sprites num arquivos .bmp para posteriormente convertê-los para .data, o conteúdo desses arquivos mostram onde cada pixel está, para ser printado.

Para rodar o jogo e testar suas funcionalidades antes de inserir na placa, foi utilizado o FPGRARS que visa fornecer um simulador de assembly RISC-V com exibição de gráficos coloridos de 8 bits e entrada de teclado, semelhante ao RARS, mas mais rápido, e na implementação final utilizamos o Kit FGPA DE1-SoC Altera, uma plataforma de design de hardware robusta construída em torno do Altera System-on-Chip (SoC) FPGA, que combina os mais recentes núcleos incorporados Cortex-A9 dual-core com lógica programável líder do setor para máxima flexibilidade de design. Seguiu-se uma metodologia de game design, ela deu o caminho e os passos necessários para que o trabalho fosse conduzido de forma organizada, sem ruídos de comunicação, evitando retrabalho e agilizando a produção.

4 Resultados Obtidos

4.1 História

Para se tornar um jogo completo e coeso, foi necessário criar uma história cativante e que fizesse sentido para o jogador. O enredo foi inspirado no jogo original, os pokemons, o professor e muitos aspectos surgiram daí, além disso, trouxe-se diversas refêrencias da própria universidade e o ambiente.

A história trata-se de um calouro que se encontra preso na cidade de OACapulco e a única forma de escapar é derrotando o cobrador do ônibus numa batalha pokemon, para isso, ele conta com a ajuda do professor Lamar para instruí-lo e dar o seu pokemon para batalhar, então o calouro precisa se aventurar na cidade atrás de pokemons selvagens até conseguir comprar o passe e lutar com o cobrador.

Foi desenvolvido uma espécie de roteiro baseado nas falas que aparecem no jogo, através disso, foi possível passar a ideia da trama e tornar o jogo mais divertido. Aqui está um trecho de exemplo do roteiro:

LABORATÓRIO
 # jogador interage com o professor lamar
Lamar: "Antes de começarmos..."
Lamar: "Ja fez o testinho hoje?"

4.2 Música e Som

Para poder tocar uma música durante o jogo foi necessário implementar uma função que não usasse o sleep entre as notas, pois o isso prejudicaria a experiência do jogador. Portanto, foi implementada uma função que ao tocar uma nota, ela armazena na ram o tempo de pausa entre uma nota e outra, o tempo em que foi tocada a nota e um contador para saber qual seria a proxima nota tocada. Tendo todas essas informações, quando a função for chamada novamente para tocar a proxima nota ela verificará se a diferença entre o tempo em que foi tocada a ultima nota e o tempo atual é maior ou igual que o tempo de pausa armazenado, caso seja, ela tocará a nota e atualizará os dados na ram, caso contrário, ela voltará a rodar o jogo normalmente. Essa função é chamada em toda iteração do loop do jogo.

4.3 Itens

O jogo possui 2 itens e um sistema de dinheiro, esses itens são essenciais para completar o jogo, um deles é a **marmita**, que serve para curar o pokemon do jogador, ela recupera 10 pontos de vida e pode ser obtida na loja por 50 créditos.

O outro item é o **passe**, que é justamente o item necessário para entrar no ônibus, o 110, e poder lutar com o inimigo final, ele custa 200 créditos e pode ser comprado com o professor Lamar. É interessante ressaltar que, se o jogador perder a batalha final, ele perderá o passe e precisará comprá-lo de novo.

O sistema do dinheiro é bem básico, a moeda é chamada de **créditos** e o jogador começa sem nenhuma e a única forma de adquirí-los é derrotando os pokemons selvagens, ele pode utilizar esses créditos para comprar os itens citados anteriormente.

4.4 Pokemons

Foi definido que o protagonista pode escolher somente um pokemon dentre 3 iniciais: Pedro Saruê, Pombo do ICC e Caramelo.

Além dos iniciais, foram implementados outros 3 pokémons: Larva do RU, Gato e Sinuca do CACOMP. Os três pokémons iniciais, a Larva do RU e o Gato podem ser encontrados nos matos espalhados pelo mapa. A Sinuca do CACOMP é pokémon do cobrado do 110 e deve ser derrotado para que o jogador ganhe o jogo. Os 6 pokemons são baseados em animais e elementos que podem ser encontrados na Universidade de Brasília, bem característicos e icônicos.

Cada pokémon possui atributos oito atributos diferentes: ataque, defesa, vida, vida máxima, level, xp, tipo, velocidade. esses atributos são usados para implementar a batalha entre pokémons. Para aumentar o level, o jogador precisa conseguir xp referente a cinco vezes o level atual. Os pokémons conseguem xp derrotando outros pokémons em batalha. Cada level aumenta os atributos de vida e vida máxima em dois e os atributos de defesa, ataque e velocidade em 1.

4.4.1 Pedro Saruê

Figura 3: *Sprite do Pokemon Pedro Saruê*

O Pedro Saruê é um pokémon do tipo Atirador com os seguintes atributos: 6 de ataque, 6 defesa, 12 de vida, 12 de vida máxima e 6 de velocidade no level 1. Possui os ataques Tapasso, Dentada e Olhada.

4.4.2 Caramelo

Figura 4: *Sprite do Pokemon Caramelo*

O Caramelo é um pokémon do tipo Lutador com os seguintes atributos: 6 de ataque, 6 defesa, 12 de vida, 12 de vida máxima e 6 de velocidade no level 1. Possui os ataques Tapasso, Patada e Olhada.

4.4.3 Pombo do ICC

Figura 5: *Sprite do Pokemon Pombo do ICC*

O Pombo do ICC é um pokémon do tipo Voador com os seguintes atributos: 6 de ataque, 6 defesa, 12 de vida, 12 de vida máxima e 6 de velocidade no level 1. Possui os ataques Tapasso, Revoada e Olhada.

4.4.4 Larva do RU

Figura 6: *Sprite do Pokemon Larva do RU*

A Larva do RU é um pokémon do tipo Venenoso com os seguintes atributos: 6 de ataque, 6 defesa, 11 de vida, 11 de vida máxima e 5 de velocidade no level 1. Possui os ataques Tapasso, Teníase e Olhada.

4.4.5 Gato do ICC

Figura 7: *Sprite do Pokemon Gato do ICC*

O Gato é um pokémon do tipo Normal com os seguintes atributos: 5 de ataque, 5 defesa, 11 de vida, 11 de vida máxima e 6 de velocidade no level 1. Possui os ataques Tapasso e Olhada.

4.4.6 SINUCA CACOMP

Figura 8: *Sprite do Pokemon Sinuca CACOMP*

O Caramelo é um pokémon do tipo Venenoso com os seguintes atributos: 6 de ataque, 6 defesa, 11 de vida, 11 de vida máxima e 7 de velocidade no level 1. Possui os ataques Prego e Capote.

4.5 MOVIMENTOS E ATAQUES

Cada pokémon possui entre 2 a 4 ataques com tipos e ações diferentes. Cada ataque possui um tipo, um nome, um poder e uma ação. O grupo implementou 2 variedades de ataques: os ataques que são dano e os ataque que mudam os atributos dos pokémons.

4.5.1 ATAQUE DE DANO

Foram produzidos 6 ataques que dão dano no pokémon inimigo: Revoada (Voador), Atirador, Patada (Lutador), Tapasso (Normal), Teníase (Venenoso) e Capote (Venenoso). Todos esses ataques possuem 40 de poder. O dano desses ataques é calculado com base na seguinte equação

$$Dano = (\frac{(\frac{2.lvl}{5}+2).Power.\frac{A}{D}}{50}+2).Type1,$$

onde lvl é level do pokémon atacante, Power é poder do ataque, A é o ataque do pokémon atacante, D é a defesa do pokémon defensor e Type1 é o quão efetivo o tipo do ataque é efetivo no pokémon defensor ($Type1 = 0.5$- se não for efetivo-, $Type1 = 1$- se tiver eficácia normal- e $Type1 = 2$- se o atauqe for muito efetivo).

4.5.2 ATAQUES QUE MUDAM ATRIBUTOS

Foram implementados 2 ataques que alteram o ataque do pokémon inimigo em 1: Olhada (Normal) e Prego (Normal).

4.6 TIPOS

Desenvolveu-se 5 tipos diferentes dentro do jogo: Normal, Voador, Atirador, Lutador e Venenoso. Cada pokemon possui o seu tipo específico e eles funcionam como no jogo original, por um sistema estilo pedra-papel-tesoura, os tipos são atribuídos aos diferentes ataques e pokemons. Foi criado esta tabela para definir como cada tipo reage contra outro. Por exemplo, se o jogador usar um ataque do tipo Voador contra um pokemon do tipo Lutador, o ataque será super efetivo, ou seja, receberá o dobro do dano base, a tabela mostra isso com o respectivo multiplicador para cada combinação:

Ataque \ Defensor	Normal	Voador	Atirador	Lutador	Venenoso
Normal	1	1	1	1	1/2
Voador	1	1/2	1/2	2	1/2
Atirador	1	2	1/2	1/2	2
Lutador	1	1/2	2	1/2	2
Venenoso	1	2	1	1	1

Tabela 1: *Tabela dos multiplicadores de dano, de acordo com o ataque e receptor.*

4.7 BATALHA

A batalha do jogo foi baseada nos jogos originais e feita de maneira muito similar a eles. Em cada turno, o jogador pode escolher realizar um ataque com seu pokémon, usar um item ou tentar fugir da batalha, caso a batalha seja com um pokémon selvagem. Caso o jogador tenha escolhido utilizar um item, ele possui a prioridade no turno e o item é usado antes da ação do pokémon inimigo. Se o jogador tenha escolhido realizar um ataque, o pokémon com maior atributo de velocidade atacará primeiro. As ações dos pokémons inimigos são escolhidas por duas funções que serão explicadas na próxima sessão.

Ao final da batalha, caso o jogador tiver ganhado, o pokémon do jogador ganha xp e é verificado se ele alcançou a xp necessária para passar para o próximo level. Além disso, o jogadar ganha 10 créditos por cada batalha ganha.

Figura 9: *Sprite do Pokemon Sinuca CACOMP*

4.8 IA

Foram implementadas duas funções para tomada de decisão dos pokémons inimigos durante a batalha: uma para os pokémos selvagens e a outra para a batalha contra o cobrador do 110. A função para os pokémons selvagens escolhe os ataques de maneira aleatória. Já na função implementada para a batalha com o cobrador, foi desenvolvida a seguinte estratégia: no primeiro turno, o pokémon ataca com o ataque prego para diminuir o ataque do pokémon do jogador, nos demmais turnos ele ataca com Capote e, quando a Si-

nuca do CACOMP chegar a menos da metade da vida pela primeira vez, o npc usará uma poção para curar seu pokémon.

4.9 SPRITES

O jogo possui uma variedade de sprites, basicamente foram divididos entre os sprites de tile (122 no total), os de imagens (7 no total)e os de caixa, (22 no total) os tiles são blocos de 16x16 pixels que foram aplicados para serem impressos na matriz do mapa, que também é subdividida em 16 partes, os sprites normais são de tamanho maior, geralmente com 60 pixels de altura, os pokemons, por exemplo, são desse tipo e por último, tem-se as sprites de caixa, que são formadas somente pelas bordas de cada caixa e com elas em conjunto com a cor de fundo, formam as caixas de texto e o menu. Estes são alguns exemplos dos tipos de sprites:

Figura 10: *Exemplos de sprites*

Depois que foram criados as sprites de tiles, (A maioria retirado do jogo original) foi necessário criar uma maneira de mapear tais tiles na matriz dentro do jogo, por isso foi desenvolvido um script que retorna a cor azul dos pixels de um arquivo .bmp de 24 bits em código hexadecimal, com isso foi possível fazer um mapa de tiles com as determinadas cores, modificando o valor azul, assim foram determinados os ponteiros para cada sprite específico.

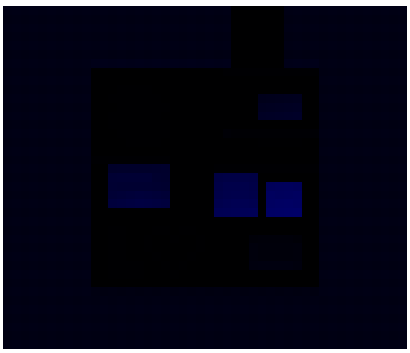

Figura 11: *Mapa de tiles do mapa principal*

Também foi desenvolvido um guia para facilitar esse tipo de conversão, nele está todos os códigos hexadecimais formados por dois dígitos da cor azul que representará algum sprite de tile específico, nele está mapeado quais blocos serão interagíveis ou refletidos também. Este é um trecho do guia que contém as sprites do ônibus:

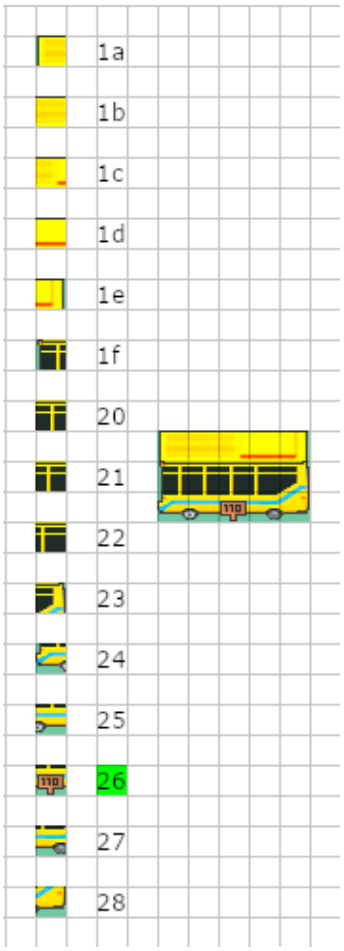

Figura 12: *Trecho do guia de tiles*

Cada pokémon possui seu sprite único e ele é impresso na tela durante batalhas. As caixas de texto servem de base para utilizar uma função do arquivo SYSTEMv21.s que mostra strings definidas, por meio disso, conseguiu-se colocar as informações no menu, as falas e as opções de escolha.

4.10 MAPA PRINCIPAL

Todos os mapas do jogo foram feitos utilizando o sistema de tiles. Esse sistema consiste em implementar mapas como tabuleiros de xadrez, nos quais o jogador só pode assumir posições fixas, as posições das tiles. Dessa forma, economizamos muita memória fazendo apenas as sprites das tiles, possibilitando-nos fazer uma mapa relativamente grande. Além disso, quando o jogador se movimenta pelo mapa, a parte do mapa renderizada na tela muda, ou seja, a renderização do mapa é dinâmica relativa à posição do jogador no mapa.

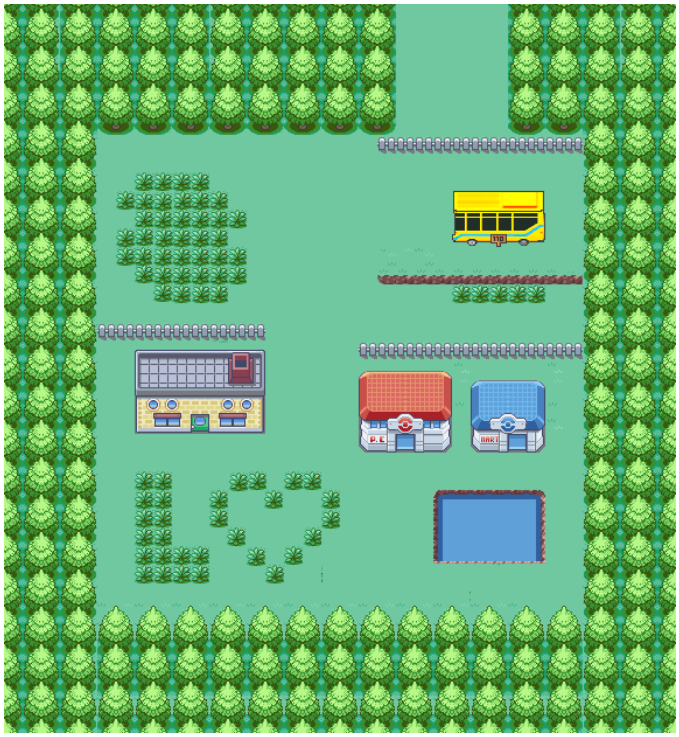

Figura 13: *Mapa principal*

4.11 Terrenos

Foram implementados diversos tipos de terrenos pelo sistemas de tiles. Dentre eles, os mais importantes são a grama alta- terreno em que os pokémons aparecem para o jogador-, a grama baixa- terreno padrão em que o jogador se movimenta por ele-, a água- terreno em que o jogador não pode andar-, o barranco- terreno em que o jogador não pode ultrapassar- e os terrenos interagíveis, como as portas e os npcs.

Figura 14: *Grama alta*

Figura 15: *Grama baixa*

Figura 16: *Água*

Figura 17: *Barranco*

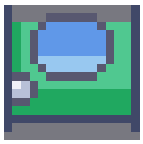

Figura 18: *Porta - Terreno Interagível*

4.12 Movimentação

O jogador pode movimentar o personagem na tela por meio das teclas 'a', 'w', 's', 'd'. A tecla 'a' movimenta o jogador para a esquerda, a 'w' movimenta o jogador para cima, a 's' para baixo e a 'd' para a direita. Assim, quando se aperta uma dessas teclas e o personagem está virado na direção desejada, a posição do jogador é aumentada ou diminuída em 1 tile dependendo da direção. Caso o personagem não esteja virado na posição do input, em vez de andar, o personagem se vira a direção em questão. Além disso, como citado na sessão Mapa Principal, a movimentação do jogador influencia na renderização do mapa. Logo, a posição do jogador é utlizada para calcular as tiles que precisam ser renderizadas.

4.13 Interiores

O jogo possui um total de 4 interiores, todos são acessíveis pelo mapa principal, eles possuem um tile que o jogador pode entrar. Todas as edificações do mapa possuem um interior, dentro deles existe um NPC em que o jogador pode interagir. Determinou-se que toda vez que o jogador entrar em um local, o recinto será impresso com um fundo preto, a movimentação e algoritmo funciona basicamente como o mapa principal, por uma matriz, e também tem o seu próprio mapa de tiles. Estes são os interiores existentes:

4.13.1 Laboratório

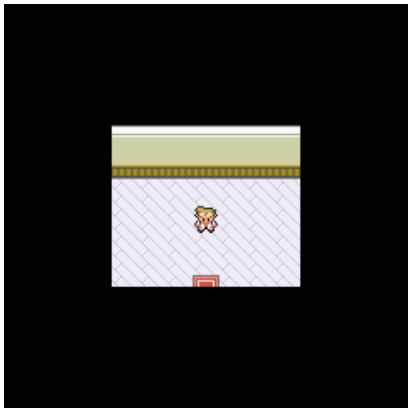

Figura 19: *Interior do Laboratório*

Dentro do laboratório você pode encontrar o professor Lamar e interagir com ele, você pode comprar o passe com ele e escolher seu pokemon incial, é possível sair pelo tapete também.

4.13.2 RU

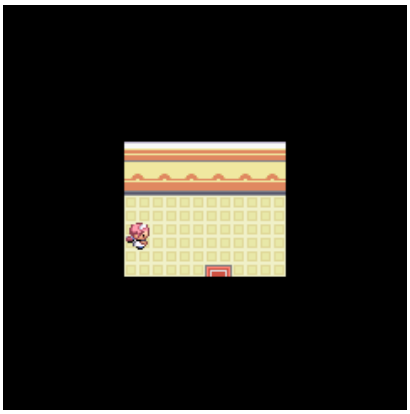

Figura 20: *Interior do RU*

Nomeado em homenagem ao restaurante universitário, aqui você pode interagir com a moça do RU e alimentar seu pokemon para recuperar seus pontos de vida.

4.13.3 Loja

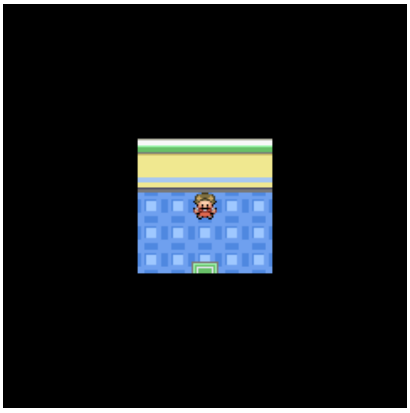

Figura 21: *Interior da Loja*

Ao entrar na loja é possível observar um vendedor e ao conversar com ele, ele oferece para vender uma marmita ao jogador.

4.13.4 Ônibus 110

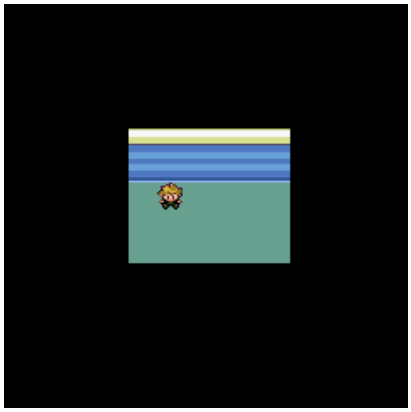

Figura 22: *Interior do Ônibus*

Este é o único interior que o jogador não pode entrar livremente e nem pode sair assim que estiver dentro, nele está o vilão final e só é possível sair do local lutando com ele.

4.14 Menus

Ao longo do jogo, aparecem diversos menus. Grande parte deles foram implementados a partir de funções que retornavam um valor de acordo com a posição de uma seta. Além disso, foi implementado um menu principal, o qual pode ser acessado clicando a tecla "Esc". Esse menu exibe informações importantes sobre o jogador e o pokémon inicial escolhido, como o level e quantidade de xp do pokémon e a quantidade de marmitas e de créditos que o jogador possui.

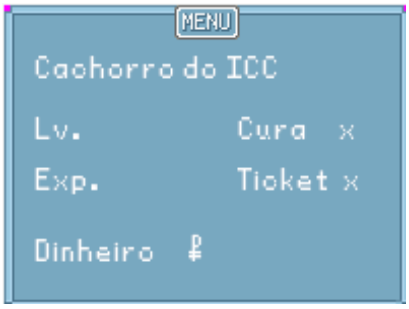

Figura 23: *Menu principal*

5 Conclusões e Trabalhos Futuros

Em suma, o grupo conseguiu alcançar o proposto, desenvolver a habilidade de programação na linguagem Assembly RISC-V e fazer um projeto para ser implementado na placa FPGA, tal projeto que concluiu com os requisitos mostrados. Ao decorrer do projeto, as práticas de programação em grupo e desenvolvimento de técnicas de jogos foram fortemente aplicadas, foi necessário muita capacidade de resolução de problemas para lidar com bugs e implementar funcionalidades novas. Os integrantes buscaram se organizar e subdividir as tarefas igualmente e coesamente para otimizar a produção, esta tarefa foi uma das mais difíceis, por isso foi necessário grande organização de cada componente.

Foram observadas diversos tipos de dificuldades e obstáculos durante a produção, o maior problema, de fato, foram os bugs e glit-

ches, o RARS é uma ferramenta muito difícil de realizar debug e bastante lenta, portanto se tornou complicado resolver os problemas do código. Outro fator que dificultou muito, foi a questão da placa FPGA possuir diferenças de otimização em relação ao FPGRARS, isso gerou erros de alinhamento, que somente ocorreram ao tentar executar na placa, ou seja, foi preciso fazer debug testando diretamente na placa, o que complicou a logistíca, pois os integrantes precisaram ir ao campus, por ser o único local que possui a placa. Ao executar na placa, foi observado que o processador ideal para executar o jogo foi o processado RISC-V Uniciclo com a ISA RV32IMF.

Apesar das dificuldades, o grupo se esforçou para trazer o melhor trabalho possível dentro do prazo estimado e se surpreendeu com a quantidade de conhecimento adquirido durante a realização do projeto, conhecimento que será utilizado ao decorrer da graduação. Acerca de trabalhos futuros, os integrantes não se sentiram motivados para produzir mais projetos em Assembly, pelo fato da linguagem ser de baixíssimo nível, porém alguns integrantes do grupo se interessaram bastante pela criação de jogos no geral.

6 Referências Bibliográficas

Referências

[1] Patterson, D.A., Hennessy, J.L. *Organização e Projeto de Computadores – A Interface Hardware/Software, 3ª edição, Editora Campus, 2005;*

[2] Professor Marcus Vinícius Lamar *Aulas e slides da disciplina Organização e Arquitetura de Computadores*

[3] Wikipédia *https://pt.wikipedia.org/wiki/PokÃ©mon_FireRed_e_LeafGreen*. Wikipédia Pokemon FireRed e LeafGreen