PARTICIPANTES

Felipe da Rocha de Brito Machado - 251000107
Moisés Gibson dos Passos - 251025966
Marcos Lopes Martins - 251001140

PREÂMBULO

Trata-se do projeto de final de semestre da disciplina Introdução aos Sistemas Computacionais (ISC) cujo propósito é a elaboração de um programa em linguagem Assembly RISC-V (*Reduced Instruction Set Computer*) para um aplicativo de entretenimento denominado "Bomberman" que deverá ser executado usando simulador FPGRars.

Durante o desenvolvimento deste trabalho colocamos em prática os conteúdos ministrados e aprendidos em sala sobre esta linguagem de programação, assim como: gerenciamento de memória, registradores e outros elementos necessários para sua compilação.

INTRODUÇÃO SOBRE A LINGUAGEM UTILIZADA

A programação em Assembly teve um papel crucial no início da história da computação, sendo uma das primeiras formas de interagir diretamente com o hardware do computador. Surgiu como uma alternativa ao código binário, utilizando mnemônicos (abreviações de palavras em inglês) para representar instruções de máquina, facilitando a leitura e escrita de código.

Assembly RISC-V refere-se à linguagem de programação de baixo nível usada para escrever códigos, que é diretamente executada pelo processador RISC-V. É uma arquitetura de conjunto de instruções ISA (*Instruction Set Architecture*) aberta e gratuita que permite a implementação e modificação de uma vasta gama de instruções; incluindo, por óbvio, as necessárias para a funcionalidade deste projeto.

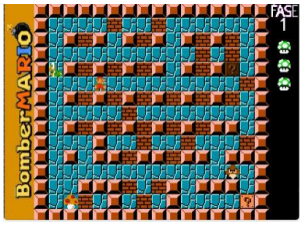
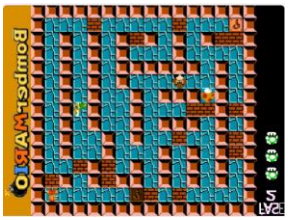
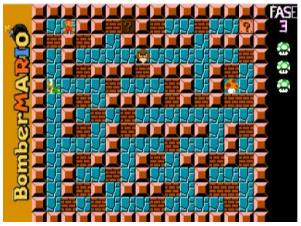
SOBRE O JOGO

Bomberman é uma série de jogos de estratégia, cujo objetivo principal é completar as fases depositando bombas em ordens e lugares estratégicos para destruir obstáculos e inimigos.

Inicialmente desenvolvido pela fabricante Hudson Soft em 1983, atualmente é produzido, desenvolvido, distribuído e comercializado pela Nintendo. Novos jogos da série são constantemente lançados. No mercado já existem mais de 60 jogos diferentes com este personagem.

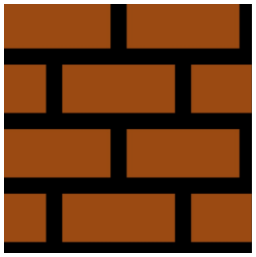
Bomberman é considerado um clássico para muitos jogadores de videogame.

ANIMAÇÕES E PERSONAGENS

labirintos

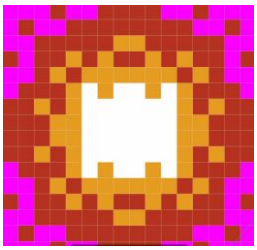
parede inquebrável



bloco “quebrável”

delimita o caminho – chão

personagem Mario



bomba

Observação: todas as imagens / sprites acima são divididas por números no código e estão representadas com comentários.

METODOLOGIAS E FUNCIONALIDADES UTILIZADAS PARA DESENVOLVIMENTO DO APLICATIVO

MÚSICA

As músicas que utilizamos neste aplicativo são composições próprias. Elas foram compostas utilizando o software FL Studio, que funciona como uma estação de trabalho de áudio digital, que possui uma interface gráfica de usuário com um sequenciador de música baseado em padrões.

Passo seguinte foi traduzir as músicas para uma linguagem apropriada para a plataforma RISC-V, em uma série de arquivos com extensão .data.

Utilizando as instruções *ecall 31* e *ecall 30* do RARS foi possível gerar as notas musicais compostas nos períodos de tempo e intervalos apropriados, bem como assegurar que todas elas sejam executadas de acordo com a ordem necessária e correta do andamento do jogo.

MOVIMENTOS E ANIMAÇÕES

Em jogos, um sprite é uma imagem bidimensional ou animação que é integrada em um cenário maior ou ambiente de jogo. Eles são usados para representar personagens, objetos, efeitos especiais e outros elementos visuais em jogos 2D e também podem ser usados em jogos 3D para otimizar o desempenho ou criar efeitos específicos.

Em essência, os sprites são a base para a criação de gráficos em jogos, especialmente nos jogos 2D, onde cada movimento ou ação de um personagem é geralmente representado por uma sequência de sprites.

No nosso caso, utilizamos os seguintes sites: https://www.spriters-resource.com/nes/supermariobros/ para todos os sprites utilizados no jogo e o Gemini pra ajudar nas imagens de vitória, derrota e início.

A animação do Mario andando consiste de 5 sprites que estão constantemente sendo alternados, um em cada frame, para dar a sensação de movimento do personagem. A animação dos inimigos segue uma lógica análoga, porém com apenas 2 frames.

Para converter arquivos em .data, nós utilizamos o aplicativo de edição *Krita*, que consegue converter um .png em uma imagem BMP Windows. Depois de converter a imagem .png pra um arquivo .bmp, usamos um programa chamado bm2oac3.exe, que converte arquivos .bmp em 3:1, necessários dentro do contexto do projeto.

Para a explosão da bomba, utilizamos um sistema que, no instante que o jogador apertar "b" no teclado, um timer de 2000 ms será ativado e um loop verifica constantemente se o tempo foi atingido ou não. Enquanto não for atingido, o jogo

prossegue normalmente, sem alterações. Todavia, assim que a bomba explode, o mapa é alterado e uma cruz em volta da bomba transforma todos os blocos quebráveis em chão.

Para verificar se houve colisão com qualquer objeto, o *game loop* constantemente avalia se o jogador se encontra dentro dos limites estabelecidos pelos mapas, que são arquivos .data com matrizes dividindo a função de cada bloco. Para isso, criamos uma função CHECK_TILE que realiza esta rotina. Se o jogador estiver tentando passar por cima de um bloco inquebrável, tal ação não é executada e um sinal sonoro á ativado, a depender do tipo de bloco.

Selecionamos um pixel x0 e y0 pra apresentar tanto nosso *banner*, quanto nosso HUD (*Heads-Up Display*) na direita. Por ser uma tela de 320x240 pixels, houve necessidade de calcular quantos "blocos" de distância (16x16 cada) gostaríamos que cada imagem tivesse. Para isso, utilizamos a função *print*, a qual felizmente foi ministrada e testada em nosso laboratório, que combinada com outras funções já utilizadas em projetos anteriores, performa uma função que recebe uma determinada largura e altura, e que se adapta melhor à estrutura do código gerado pelo executável bitmap2oac3.exe.

RESULTADOS OBTIDOS

Quando recebemos as orientações sobre as especificações mínimas deste jogo, já sabíamos que estávamos frente à uma tarefa extremamente desafiadora; porém, ao mesmo tempo, instigante. A linguagem utilizada neste trabalho é bastante complexa e os detalhes e variantes para viabilizar cada um dos quesitos foram inúmeros. Porém, pouco a pouco, eles foram superados e, como resultado, chegamos a soluções bastante satisfatórias e compreensivas para o resultado final do jogo.

CONCLUSÃO

O desenvolvimento deste aplicativo nos permitiu colocar em prática os conceitos teóricos desenvolvidos em sala, assim como as atividades de laboratório, da disciplina de Introdução aos Sistemas Computacionais (ISC). Mais especificamente no que tange a utilização da linguagem de baixo nível Assembly RISC-V, a qual foi base para a implementação deste jogo.

AGRADECIMENTOS

Deixamos registrado os nossos melhores agradecimentos ao monitor José Edson.

BIBLIOGRAFIA E REFERÊNCIAS

www.wikipedia.com
https://www.spriters-resource.com/nes/supermariobros/
https://krita.org/en/
www.resizepixel.com

https://www.youtube.com/watch?v=CZ9Pu9Usk5o
https://online.oldgames.sk/play/nes/bomberman/6762
Material de apoio disponível no site https://aprender3.unb.br/ dentro da disciplina Introdução aos Sistemas Computacionais