# Projeto Helltaker

André Dantas Ferreira da Silva, Breno Costa Avelino Lima, Oseias Romeiro Magalhães

Brasil, UnB - Universidade de Brasília, Dept. de Ciência da Computação

## Resumo

O objetivo deste projeto é aplicar os conhecimentos adquiridos em Introdução aos Sistemas Computacionais (ISC) na programação de um jogo semelhante a Helltaker utilizando a arquitetura RISC-V. RISC-V é uma ISA (Instruction Set Architecture ou arquitetura de conjunto de instruções) que foi desenvolvida originalmente na Universidade da Califórnia. Ela tem como principal objetivo desenvolver um processador de alto desempenho e de código aberto, que possa ser utilizado no desenvolvimento de aplicações e produtos de uso geral.

Aqui será descrito todo o processo e metodologia utilizados no desenvolvimento do jogo, como foram feitos os loops de gameplay, como foi feita a renderização de imagens na tela, as músicas e a comunicação com o teclado.

## 1 – Introdução

Programação permite que o computador seja utilizado das mais variadas formas como realizar cálculos matemáticos, realizar tarefas repetitivas, ver e editar imagens, sons entre várias outras coisas. Para saber o quanto uma pessoa sabe sobre determinada linguagem é importante que sejam testadas suas habilidades na maior parte dessas possibilidades. A escolha de um jogo serve justamente para isso, pois em um jogo é preciso fazer cálculos matemáticos, tarefas que se repetem (game loop), reproduzir imagens e sons e ter uma comunicação com o teclado para que o jogador possa interagir.

O jogo que foi escolhido para ser reproduzido em linguagem assembly do RISC-V é o Helltaker. É um jogo simples que consiste em levar seu personagem de um ponto a outro do mapa com um número finito de movimentos, tendo pelo caminho obstáculos como pedras, inimigos e espinhos, que fazem o personagem ter que gastar mais movimentos para realizar determinada tarefa. A figura 1 abaixo mostra uma imagem do jogo presente na página dele na loja steam.


Figura 1: Imagem do jogo na loja steam.

O código foi feito pelo programa RARS feito em Java. O programa simula a operação de um processador RISC-V. Também foram utilizadas as ferramentas Bitmap Display, para renderização das imagens, e Keybord and Display Simulator para fazer a interação com o

teclado. As músicas foram feitas utilizando ecalls do próprio RARS.

Na seção metodologia será explicado como a equipe fez a programação dos diferentes aspectos do jogo, assim como as dificuldades que apareceram durante o processo.

# 2 - Metodologia

Começando com as imagens utilizadas no jogo. Para obtê-las foram utilizados dois programas: o paint.net e um programa fornecido pelo professor. O paint.net foi utilizado para editar as imagens, principalmente sua resolução e suas cores, e também foi utilizado para salvar as imagens no tipo de arquivo .bpm de 24 bits. Com esse arquivo o outro programa converte essa imagem para outro tipo de arquivo .data, esse que pode ser utilizado dentro do RARS.

A primeira parte do jogo que aparece ao ser iniciado é o menu. Para fazer o menu primeiro foi renderizada nos frames 0 e 1 a imagem da figura 1 e junto dela uma frase explicando como acessar o jogo. Para imprimir essas imagens que ocupam a tela inteira simplesmente foi percorrido todos os endereços dos pixels tanto do frame 0 quanto do 1 utilizando o arquivo .data da imagem. A frase foi impressa na tela utilizando a ecall 104, para utilizá-la foram incluídos no programa dois arquivos: “MACROSv21.s” e “SYSTEMv21.s”.

A partir daí foi feito um loop onde acontecem dois eventos. O primeiro é a música que foi feita à mão e reproduzida utilizando as ecalls 31 que reproduz uma nota e a 32 que dá um delay entre notas, quando a música acaba ela é repetida. O segundo é a verificação do teclado que se determinada tecla é pressionada ele sai desse loop e apaga as imagens dos frames iniciando o game loop.

Antes de explicar o game loop é preciso explicar que para renderizar as imagens que são menores que a tela foi utilizada uma técnica que consiste em pegar o primeiro endereço do pixel (no caso o do canto superior esquerdo) e o último endereço (canto inferior direito) e ir renderizando a imagem ate chegar no canto direito, somando então ao valor total de pixels na horizontal menos o tamanho da imagem para que o próximo endereço esteja no lugar certo.

O game loop funciona com uma sequência de interações desencadeadas pelo código digitado no teclado. Assim que o jogo é iniciado, o programa lê um arquivo .data e imprime o mapa da fase no bitmap display. O mapa do jogo não se encaixa perfeitamente no display retangular de 320x240 do RARS, portanto o programa deixa propositalmente uma área preta maior à direita da tela. Assim que o mapa está pronto, o game loop inicia de fato. Primeiramente, carregamos o endereço de memória do Keyboard MMIO (ferramenta de leitura do teclado do RARS), e verificamos se algum input foi detectado. Caso nada tenha sido digitado, o programa volta e verifica novamente. Caso algo seja digitado, uma sequência de condicionais verifica qual tecla foi digitada e aciona uma ação correspondente. As teclas A, W, S, D fazem o personagem se mover no espaço. Além das teclas de movimentos (W, A, S, D) o jogo contém também o procedimento de reiniciar o jogo, que é chamado ao pressionar a tecla “R” que reinicia a fase, retornando o personagem para o início e limpando as alterações feitas no mapa. O procedimento também é reaproveitado para o contador de movimentos, que ao ultrapassar o limite programado para a determinada fase

chama o procedimento de reiniciar. Qualquer outra tecla pressionada não aciona ação alguma no jogo.

Depois de lida a tecla, caso seja uma tecla de movimentação, o programa analisa o quadrado de destino da movimentação para avaliar qual ação será tomada. Caso o quadrado à frente esteja livre, as coordenadas do personagem são atualizadas somando um número positivo ou negativo na world correspondente, salva seu novo valor, apaga o personagem da posição anterior e substitui por um quadrado de chão. Caso o quadrado a ser acessado seja uma caixa, o personagem permanece no lugar e a mesma interação anterior segue para a caixa. Verifica-se se o quadrado à frente da caixa está vago; caso esteja, a caixa se move e o ocupa, caso contrário, nem a caixa nem o personagem saem do lugar e um ponto de movimento é consumido. A mesma lógica se aplica para inimigos, espinhos e paredes do mapa de acordo com suas mecânicas específicas. Após cada ação, o game loop se reinicia aguardando uma nova entrega do teclado.

Cada elemento no mapa é estabelecido segundo uma matriz de números armazenada no programa. 0 representa buracos, 1 representa chão livre, 2 representa o personagem, etc. Em cada loop de interação, o programa confere com essa matriz qual o elemento correspondente que se deseja acessar, e toma a ação necessária. A posição do personagem nessa matriz é fornecida no começo do programa na forma de uma word que corresponde às coordenadas x e y iniciais. A cada loop, caso a posição do personagem mude, essa word é atualizada para a posição do personagem ser diretamente acessada no próximo loop.

Infelizmente não foi possível implementar as interações do personagem com os elementos do mapa como caixas, espinhos e inimigos por diversos motivos, mas principalmente falta de familiaridade com programação de algoritmos em assembly e controle do RARS. Tivemos bastante dificuldade em mapear os elementos da matriz para o bitmap display, e implementar a lógica básica do jogo tomou muito mais tempo do que o grupo inicialmente imaginava.

# 3 – Resultados Obtidos

Tivemos como resultado uma simples versão do jogo Helltaker em RISC-V usando RARS. O jogo inicia com o menu com trilha sonora, que ao pressionar “f” inicia a fase. Ao iniciar a fase é renderizado o mapa, os elementos do cenário e o personagem principal em um ponto de origem. O personagem é capaz de se mover nas quatro direções pressionando A, W, S ou D ou reiniciar a fase apertando R caso assim deseje. Assim, iniciado a fase, o jogador deve chegar ao destino considerando o limite de movimentos e os elementos do cenário.

# 4 - Conclusão

Utilizando das ferramentas do RARS o código foi capaz de renderizar imagens, músicas e fazer a comunicação com o teclado. É possível mover o personagem na tela e reiniciar a fase, contudo faltaram funcionalidades a serem implementadas.

Apesar de todas as dificuldade e problemas que surgiram, conseguimos finalizar o essencial para o funcionamento do jogo.

# Referências


[1] https://store.steampowered.com/app/1289310/Helltaker/

[2] https://vanripper.itch.io/helltaker

[3] https://www.embarcados.com.br/fe310g-microcontrolador-open-source-estrutura-basica-risc-v/

[4] Harris, Sarah L. & Harris, David M., Projeto Digital e Arquitetura de Computadores, Morgan Kaufmann, 2015