Pac-Man VS ISC

Vinícius Regadas Sanguinetti Montezuma, Vítor Matos Guimarães e Ithalo Junio Medeiros De Oliveira
University of Brasília, Dept. of Computer Science, Brazil

Abstract

Este trabalho apresenta uma releitura do jogo ”Pac-Man”, feito em RISC-V Assembly por meio do simulador RARS para a matéria Introdução aos Sistemas Computacionais no segundo semestre de 2022, ministrada pelo professor Marcus Vinicius Lamar. A ISA (Arquitetura dde Conjunto de Instruções) RISC-V, desenvolvida pela Universidade de Berkley, na Califórnia, tem como principio ser uma arquitetura gratuita e de código e colaboração aberta.

1 Introdução

O jogo Pac-Man, lançado 22 de maio de 1980 pela empresa desenvolvedora e publicadora japonesa de jogos eletrônicos, Namco, é considerado por muitas pessoas como um dos jogos mais famosos do mundo, tendo a sua versão ”Doodle”, que é disnponível no Google, sendo ocnsiderado o quinto jogo mais jogado do mundo. Nele, o Pac-Man circula pelo mapa ”comendo” pastilhas e fugindo de fantasmas. Quando o Pac-Man ”come” uma pastilha grande ele ganha super-poderes e consegue comer os fantasmas. Originalmente, disponível em Arcades e outras plataformas de jogos eletrônicos, esse trabalho visa criar uma adapatção para RISC-V.

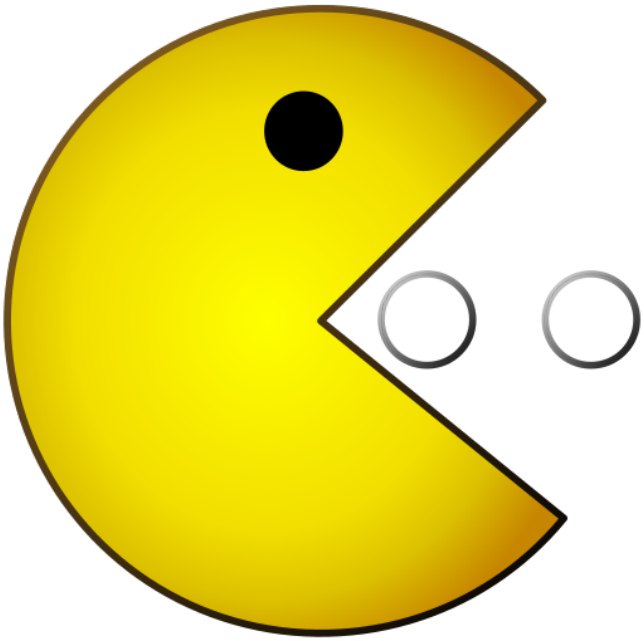

Figure 1: O Pac-Man

2 Metodologia

2.1 Movimentação básica do personagem

O funcionamento do jogo funciona na base de um game loop atualizando infinitamente. Dessa forma, a cada instância do loop é analisada a memória do KDMMIO do RARS, que verifica a inserção de algum input do jogador, caso o valor da memória seja 0, o game loop é acionado novamente. Em casos de o bit mais significativo ser diferente de 0, é averiguada qual foi a tecla pressionada, e se for percebido a presença de alguma das teclas de movimento (w, a, s, d), a função do Bitmap Display é utilizada para exibir o personagem na tela, onde o mesmo se move para a posição desejada, atualizando sua sprite lá e “apagando” sua sprite na posição antiga ao a substituir por uma imagem da mesma cor do cenário.

2.2 Matrizes

O sistema de matrizes foi o principal pilar para lidar com a movimentação do Pac-Man e suas interações com seus arredores. Apesar da representação gerada no Bitmap Display não interagir diretamente com as matrizes, as mesmas foram moldadas se baseando inteiramente em um grid de 8x8 pixeis aplicado nos mapas utilizados, sendo cada tipo de estrutura presente no grid representado por um número único. Dessa forma, a matriz utilizada de 28x31 é proporcional e representa perfeitamente os dados gráficos.

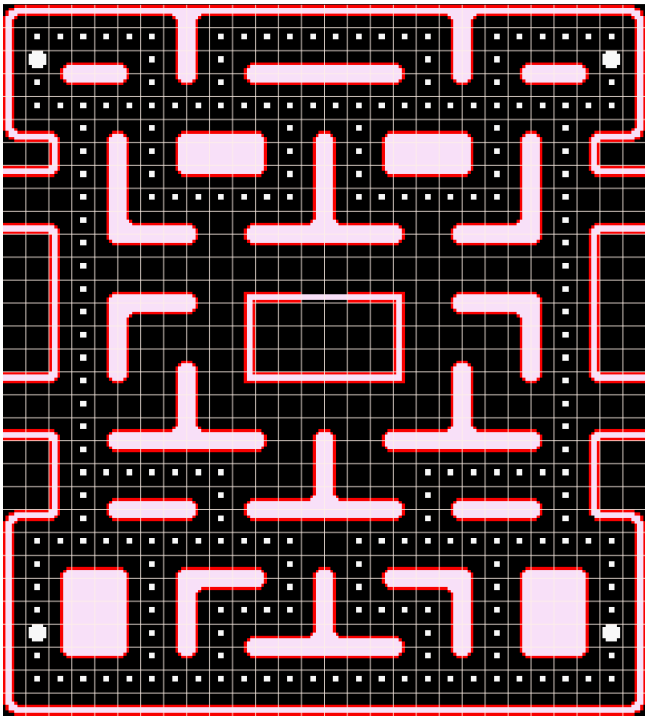

Figure 2: Grid 8x8 do primeiro mapa do jogo

Já que a representação gráfica e a representação lógica não interagem entre si, é necessário lidar com ambas ao mesmo tempo ao movimentar o personagem, atualizando agora não somente o personagem no Bitmap Display mas também a posição do mesmo na matriz. Após a atualização de variáveis, é realizado seus armazenamentos em posições de memórias alocadas na seção .data do código, para assim serem lidas futuramente e o processo se repetir a cada game loop realizado.

2.3 Sistema de Colisão

Utilizando o desenvolvimento das matrizes, o sistema de colisão foi simplificado e pôde ser implementado facilmente. Anteriormente, a posição do Pac-Man era atualizada ao ser detectado um input pelo KDMMIO, que apesar de funcional, não levava em conta a colisão entre os diversos tipos de fatores no mapa. O sistema de colisão se encaixa no centro desse processo, de forma que ao pressionar uma tecla de movimento (w, a, s, d), o endereço de pretensão do jogador é obtido através da tecla pressionada, no caso de ser pressionado uma tecla como ‘w’ seria obtido o endereço do personagem + 1, já no caso de uma tecla como o ‘s’, seria o endereço do personagem + 28 (quantidade de números em 1 linha). Com o endereço desejado, é lido o valor presente e após uma série de condicionais para cada número é decidido o que fazer em relação àquele movimento, se ele for um movimento válido a posição do Pac-Man é atualizada, caso contrário o mesmo permanece no lugar.

2.4 Movimentação avançada do personagem

A transição da movimentação básica para a movimentação avançada foi um dos processos mais demorados dentre toda a realização do jogo. Nessa parte foram realizadas as funções de se movimentar automaticamente, de armazenar o movimento desejado

até que o mesmo seja possível e então assim o praticar, a mudança de sprites interativa, e por último a movimentação de pixel em pixel. A movimentação automática teve que ser implementada através de uma pausa no game loop, porém somente na parte tangente à movimentação do personagem, de forma que o resto não fosse afetado, assim a melhor forma encontrada de implementar tal condição foi através do syscall 30 do RARS, que retorna o tempo dado desde uma data específica. Dessa forma, ao checar essa função a cada loop realizado, quando os bits mais significativos retornados fossem iguais à zero o personagem iria se mover.

```
li a7, 30 # syscall de tempo
ecall

slli s1, a0, 28
srli s1, s1, 28
# seleciona os últimos bits

beqz s1, MOVIMENTACAO
# se = 0 -> se movimenta
```

O armazenamento do próximo movimento foi dado através dos inputs do teclado, que a partir desse momento serviriam somente para isso. Para a realização disso, foi preciso criar duas variáveis que armazenam tanto a direção que o Pac-Man está se movendo atualmente quanto a direção que não é possível ser realizada no momento, mas foi recebida pelo input. No loop, é verificado se há algum valor armazenado na variável de próximo movimento, caso a mesma seja 0 o jogo continua executando a movimentação atual, caso contrário, através do sistema de colisões, é analisado se a direção desejada se tornou válida e assim é decidido se o personagem deve trocar de direções.

```
la t0, PROXIMO_MOVIMENTO
lw s1, (t0)

beqz s1, JUMP_MOVIMENTO_ATUAL
# se s1 for = 0

jal s8, PRE_CHECK_MOVIMENTO
# checa o se o movimento é válido

la t0, PROXIMO_MOVIMENTO
lw s1, (t0)
sw zero, (t0) # zera o conteudo
jr s1 # realiza o movimento
```

Com a função de pausa criada no loop, a mudança de sprites para gerar a sensação do Pac-Man abrindo e fechando a boca foi implementada, de forma que toda vez que syscall 30 retornasse 0, sua sprite atual seria trocada pela próxima na fila ordenada especialmente para criar a animação. Outro recurso implementado nas sprites foi a inversão de imagem do personagem baseada no sentido que o mesmo estivesse indo, para isso foi necessário armazenar juntamente com os movimentos a sprite específica para aquela direção a cada input do jogador, sendo essa a utilizada a cada loop para a representação gráfica.

Para finalizar a movimentação avançada, foi desenvolvida a movimentação de pixel em pixel do Pac-Man. Como a colisão está baseada no grid 8x8, era necessário que o jogador se encontrasse no meio de uma divisão desse grid para que a colisão pudesse ser avaliada. Tendo isto em vista, em todo game loop ocorrido era verificado se o personagem estava centralizado, através de uma subtração da posição do bitmap atual com a posição inicial, tanto no eixo X como no Y. Caso ambas as subtrações fossem divisíveis por 8, o Pac-Man estaria centralizado e a colisão seria efetuada. Esse processo foi um grande avanço para a fluidez do personagem, já que nas versões iniciais a movimentação avançava 8 pixeis por vez para facilitar a colisão, o que gerava uma impressão de teleporte nos mais simples deslocamentos.

2.5 Menu inicial e música

O menu foi representado por uma capa personalizada do jogo para a matéria de ISC, juntamente com uma música que utiliza a syscall 31 (MIDI) do RARS, que intercalada com a syscall 32 (Sleep), apresenta uma melodia com variadas notas ao decorrer do tempo. O jogador pode iniciar o jogo a qualquer momento, sendo apenas necessário pressionar a tecla Enter, que ao ser acionada interrompe a música imediatamente e introduz um efeito sonoro de sino, começando assim a gameplay. A melodia da música foi selecionada através do site HookTheory, e implementada baseada no arquivo midi.s disponibilizada pelo Professor em seus programas que exemplificam o funcionamento do RARS.

Figure 3: Menu inicial do jogo

2.6 Menu in-game com informações do jogo

O menu in-game utilizou as macros disponibilizadas pelo Professor para exibir as informações essenciais da gameplay, tal como a pontuação, a quantidade de vidas, o score máximo e as frutas coletadas. Assim, através das syscalls 101 e 104 das macros, foi possível apresentar os principais dados escolhendo suas posições x,y no Bitmap Display, o que possibilitou separar as strings que indicam as informações e seus respectivos valores armazenados através de toda a tela do jogo, gerando um menu simétrico e funcional.

Figure 4: Menu in-game

2.7 Condição de vitória e derrota das fases

Baseando-se no sistema de colisão, o choque com as comidas pequenas e grandes geram uma pontuação e a quantidade capturada das mesmas é armazenada em uma variável. No game loop, é constantemente avaliado se a quantidade obtida de comidas é igual à quantidade total de comidas do mapa, algo que também atualiza a cada mudança de labirinto, caso seja, é realizada a troca do mapa, atualizando as variáveis específicas para cada um deles, como posição x,y do Pac-Man no Bitmap Display, sua posição na matriz e entre outros. Dessa forma, se for detectado que o mapa que o jogador se encontra é o último, ao realizar a coleta de todas as co-

midas, ao invés de ser transferido para a próxima fase será feito um redirecionamento para a tela de vitória do jogo, que poderá terminar o jogo através da tecla Enter.

Figure 5: Tela de vitória

A condição de derrota, apesar de funcionar diferente da de vitória, também opera na base da colisão. Entretanto, essa por sua vez se influencia pela quantidade de vidas do jogador, que começa seu jogo com 3 delas e perde 1 a cada contato que tem com algum dos fantasmas presente nos mapas. Caso o Pac-Man esteja na sua última vida, que pode ser checada através do menu in-game, e colida com um de seus inimigos a tela de derrota é apresentada, que semelhante à de vitória, pode finalizar o jogo através da tecla Enter.

Figure 6: Tela de derrota

3 Resultados Obtidos

3.1 Gameplay

Apesar de não estar completo, a experiência da gameplay lembra bastante a inspiração original e possui elementos o suficiente para se tornar aproveitável. Além disso, dentro das limitações apresentadas pelo Fpgrars o jogo consegue performar em uma qualidade o suficiente para que a jogabilidade não seja prejudicada. Dito isso, alguns elementos relevantes fizeram falta na versão final do jogo para que ele se tornasse mais completo como um todo, como por exemplo:

- Movimentação dos fantasmas com IA
- Implementação das frutas
- Possibilidade de recomeçar o jogo e obter score máximo
- Capacidade de matar os fantasmas ao comer a comida grande
- Mudança da velocidade conforme o passar da fase

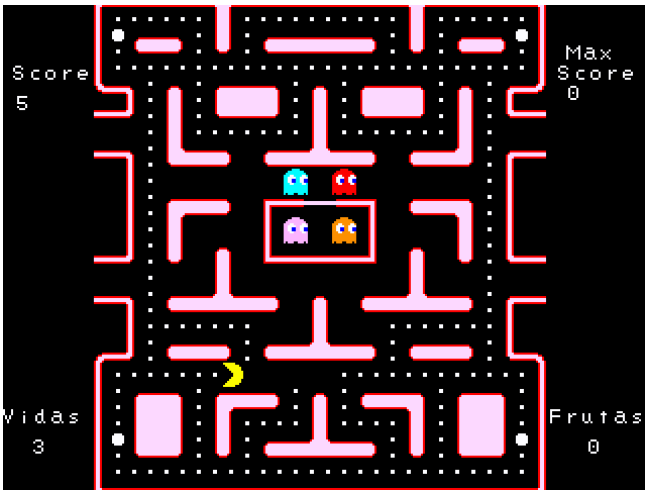

Figure 7: Imagem da gameplay

3.2 Problemas encontrados

Devido à fraca performance do RARS, o jogo se mostrava inconsistente ao rodar dentro da plataforma, constantemente crashava e ao implementar a função de parada no game loop o simulador se tornou impossível de ser utilizado, tendo que ser migrado definitivamente para o Fpgrars, o que dificultava o debug do projeto e a resolução de certos problemas, já que o mesmo não possuia todas as ferramentas do RARS nesse aspecto.

Algo que também foi notado durante o desenvolvimento foi a limitação que a função de parada causava em relação aos inputs do KDMMIO. Devido ao fato de o input do teclado só ser lido quando o syscall 30 retorna 0, existem vezes que o input sofre um pequeno delay, já que tem que esperar até a próxima exceção que retornar 0.

Em relação às sprites, essas também foram afetadas devido às limitações do Bitmap Display, por causa da proporção de 320x240 da tela, o mapa teve que ter suas partes inferior e superior cortadas e por isso a distância entre o Pac-Man e a parede nessas áreas foi afetada. Além disso, devido à necessidade das sprites serem múltiplas de 4 por causa de como o conteúdo é escrito na tela, algumas sprites tiveram que ser cortadas, como as dos fantasmas, ou achatadas e manipuladas, como a do personagem principal, para caberem nesse parâmetro, o que gera uma sensação estranha quanto ao Pac-Man, especialmente quando este se encontra indo para cima ou para baixo.

4 Conclusão

Este relatório demonstrou o processo de criação de um jogo utilizando a linguagem Assembly, no simulador RARS, que utiliza a arquitetura RISC-V.O trabalho ajudou aos integrantes do grupo a entender melhor o funcionamento de um jogo e a introduzi-los aos sistemas computacionais, já que foi a primeria vez em que a maioria do grupo teve contato com esse tipo de projeto. Com os conhecimentos aprendidos em sala de aula, foi possível usar as diversas intruções do Assembly junto com as ferramentas disponíveis no RARS para criar um jogo, o que era inimaginável para qualquer um dos integrantes no início do semestre. Por fim, foi possível notar que, mesmo sendo uma linguagem complexa, é possível utilizar diretamente a linguagem de montagem para criar uma vasta quantidade de programas.

References

LAMAR, M. V. Aulas de ISC.

MONITORES. Video aula dois monitores.

RISC-V. History of RISC-V. https://riscv.org/about/history [Accessed in 02/07/2023].

WIKIPEDIA. List of most-played video games by player count. https://en.wikipedia.org/wiki/List_of_most-played_video_games_by_player_count [Accessed in 02/07/2023].

WIKIPEDIA. Pac-man. https://pt.wikipedia.org/wiki/Pac-Man [Accessed in 02/07/2023].