The Way of the Exploding Fist

Pedro Henrique dos Santos Vitor Manuel

Marcelo Amorim

University of Brasília, Dept. of Computer Science, Brazil

Abstract

Esse projeto teve como objetivo desenvolver um jogo a partir dos estudos na matéria de Introdução aos Sistemas Computacionais. O conjunto de instruções (ISA - Instruction Set Architecture) utilizados no desenvolvimento é o RISC-V. Segundo Pereira (2017), ”a iniciativa RISC-V foi criada com o objetivo de desenvolver um processador de alto desempenho e código aberto, que possa ser utilizado no desenvolvimento de aplicações e produtos de uso geral. RISC-V é uma ISA (Instruction Set Architecture ou arquitetura de conjunto de instruções) desenvolvida originalmente na Universidade da Califórnia”. Neste artigo ficará descrito toda a metodologia de desenvolvimento do jogo, desde a definição das macros, loops que fazem a dinâmica do jogo, renderização das sprites etc. Assim, seguiremos as seguintes etapas neste artigo: introdução, metodologia, resultados, discussão e referências.

1 Introdução

Jogos digitais tem sido amplamente usados para fins de aprendizagem. Por meio das funcionalidades de um jogo é possível desenvolver diversos conceitos de programação. Durante o desenvolvimento do jogo, ficou muito marcado, por exemplo, a importância dos loops para manter a dinâmica do jogo. Também foi muito rico trabalhar com as imagens e aprender a printar diferentes sprites, gerando sensação de movimento. Tendo como referência o The RISC-V Instruction Set Manual, um dos objetivos da criação dessa arquitetura foi: ”Be simple to subset for educational purposes and to reduce complexity of bringing up newimplementations”. Desse modo, o desenvolvimento do jogo no contexto de linguagem RISC-V assembly, possibilitou diversos ganhos de aprendizagem no processo.

Figure 1: O que é a figura

A Figura 1 mostra o jogo The Way of the Exploding Fist [Altera Corporation].

Sobre o jogo

Na seção 2 será apresentada a metodologia utilizada. A seção 3 apresenta os resultados obtidos. A seção 4 conclui este trabalho.

2 Metodologia

1) A divisão de arquivos no desenvolvimento ficou da seguinte maneira:

1.1) game.s: 0 arquivo game.s é onde está o centro da aplicação. Nele foi colocado as dependências explicadas a seguir (animations-player1.s, animations-player2.s, GAME-MACRO.s, sprites.s). O programa inicia com o loop do menu e, posteriormente, o loop do jogo (GAMELOOP). No GAMELOOP fica definido todas as teclas de controle do player 1.

1.2) animations-player1.s + animations-player2.s: Aqui ficaram todas as macros ligadas aos movimentos do Player 1 e do Player 2 em seus respectivos arquivos.

1.3) GAME-MACROS.s: As game macros foram macros relacionadas ao jogo de forma geral e ficaram disponíveis para uso não só relacionada a um player.

1.4) sprites.s: Nesse arquivo ficaram todos os 'includes' relacionados às imagens em formato '.data'.

2) Utilizamos o github para versionar o código e otimizar o trabalho em grupo:

2.1) Não seguimos um padrão de versionamento, como Gitflow. Como se trata de um projeto menor, alguns commits foram diretos na branch main, tendo em vista não ter muita necessidade de ter adotado branchs develop etc. Adotamos branchs com identificadores tipo features (feat/) e fix (fix/) apenas como forma de organização do projeto. Todas as funcionalidades desenvolvidas já tiveram o merge request aceitos e estão unificadas com a branch main.

2.2) Repositório do trabalho: https://github.com/MarceloAmorim25/RISC-V-Assembly-game

3) Utilizamos o RARS como ambiente de desenvolvimento integrado (IDE) e o Visual Studio Code para edição do README.md

4) Para a renderização das imagens, foi necessária a ferramenta Bitmap Display, presente no RARS.

5) Para interagir com o teclado, Keyboard and Display MMIO Simulator.

6) As principais fontes de pesquisa foram as aulas ministradas na matéria de Introdução aos Sistemas Computacionais e os materiais de apoio.

7) Todas as funcionalidades foram comentadas para fins de documentação dos programas.

2.1 Arquitetura RISC-V

Exemplo de subseção. A arquitetura RISC-V [Patterson and Hennessy 2005] surgiu como uma forma de alternativa à dependência de padrões definidos apenas por empresas que tivessem com maior força no mercado de desenvolvimento de processadores. Um conjunto de padrões de instruções aberto veio como uma alternativa de se manter esses padrões de forma independente. Segundo o The RISC-V Instruction Set ManualVolume I: Base User-Level ISA Version 1.0, um dos objetivos da criação desse conjunto de instruções é "Provide a realistic but openISA that captures important details of commercial general-purpose ISA designs and that is suitable for direct hardware implementation.". Esse documento de marco inicial é aberto e pode ser consultado em https://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.pdf.

Um pouco mais sobre a história do RISC-V pode ser encontrada aqui https://riscv.org/about/history/.

Segundo o artigo presente em https://live-risc-

v.pantheonsite.io/wp-content/uploads/2016/02/EECS-2016-6.pdf, ”Since the instruction set architecture (ISA) is unrestricted,organizations can share implementations as well as open source compilers and operating systems.Designed for use in custom systems on a chip, RISC-V consists of a base set of instructions calledRV32I along with optional extensions for multiply and divide (RV32M), atomic operations(RV32A), single-precision floating point (RV32F), and double-precision floating point (RV32D).”. Essa passagem reforça a tese inicial de que o RISC-V surgiu como forma de existir uma arquitetura comum no mercado, não dependendo de empresa A ou B somente.

Fica claro que o RISC-V é uma interface, e não uma implementação específica, como dito em https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.pdf: ”Note that an ISA is really an interface specification, and not an implementation. There are three types of implementations of an ISA:1.Private closed source, analogous to Apple iOS.2.Licensed open source, like Wind River VxWorks.3.Free, open source that users can change and share, like Linux”.

2.2 Simulador/Montador RARS

Exemplo de subseção. O Rars *RISC-V Assembler, Simulator, and Runtime* [Vollmar and Sanderson 2005], segundo o seu repositório no github: ”RARS, the RISC-V Assembler, Simulator, and Runtime, will assemble and simulate the execution of RISC-V assembly language programs. Its primary goal is to be an effective development environment for people getting started with RISC-V.”. Esse repositório pode ser acessado no link - https://github.com/TheThirdOne/rars.

3 Resultados Obtidos

Apresentar aqui os resultados obtidos, telas e link para vídeos e comentários.

Os resultados obtidos foram a seguinte lista de funcionalidades:

- Andar para a direita (d) - Andar para a esquerda (a) - Chute (c) - Pulo central (w) - Pulo esquerda (q) - Pulo direita (e) - Rolamento (z) - Próximo nível (n) - Nível anterior (b)

No repositório do projeto, as funcionalidades possuem maior detalhamento: https://github.com/MarceloAmorim25/RISC-V-assembly-game

4 Conclusão

Este trabalho apresentou um jogo desenvolvido em linguagem RISC-V assembly. Utilizamos esse conjunto de instruções Open Source estudado na matéria de Introdução aos Sistemas Computacionais - UnB para desenvolver o jogo.

Referências:

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

References

ALTERA CORPORATION. DE2-70 development board. http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html [Accessed in 10/05/2012].

PATTERSON, D. A., AND HENNESSY, J. L. 2005. *Organização e projeto de Computadores: A Interface software/hardware, 3ª edição*. Elsevier.

VOLLMAR, K., AND SANDERSON, P. 2005. A MIPS

assembly language simulator designed for education. *Journal of Computing Sciences in Colleges 21*, 1, 95–101.