Damas 3D em RISC-V - Projeto final de OAC

Bernardo Costa Nascimento 14/0080279[1*] Cristiano Cardoso 15/0058349[2†]
Frederico de Paiva Lenza 14/0082221[3‡] João Pedro Silva Sousa 15/0038381[4§]
Vitor Moraes Dellamora 16/0056802[5¶]

[1]Universidade de Brasília, Departamento de Computação, Brasil

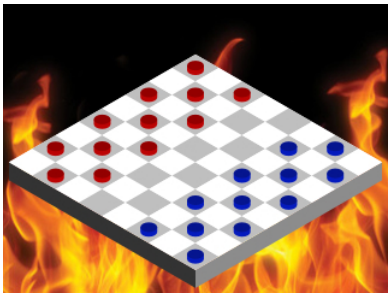

Figura 1: O tabuleiro de damas. Isométrico.

RESUMO

We introduce a custom version of the Checkers game as a proof of concept for our in-house developed RISC-V processor. A game is a complex program that typically requires most of a hardware targeting general purpose computing as known as personal computers (PCs). In this article, we demonstrate the correct execution of our game, thus the potential usage of graphics, sound and IO using an Altera DE1-SoC board and the RISC-V processor.

Keywords: Organização e Arquitetura de Computadores, RISC-V, Assembly, Damas, Universidade de Brasília.

1 INTRODUÇÃO

No primeiro semestre de 2018, na disciplina de Organização e Arquitetura de Computadores (OAC), foi decidido que a arquitetura de processadores utilizada para que a matéria fosse lecionada seria a RISC-V. Com isso, os alunos aprenderam a fazer processadores uniciclo, multiciclo e pipeline e utilizar um dos três para desenvolver, como projeto final da matéria, um jogo de damas em três dimensões feito na linguagem Assembly utilizando a ISA (*Instruction Set Architecture*) do RISC-V. Este artigo é, portanto, uma contribuição às pesquisas sobre processadores RISC-V e apresenta, detalhadamente, a fundamentação teórica e técnica, metodologia e resultados obtidos do projeto.

2 FUNDAMENTAÇÃO

Nesta seção, serão abordados quais fundamentos a equipe fez uso (teoricamente e tecnicamente) para desenvolver este projeto de jogo.

*e-mail: bernardoc1104@gmail.com
†e-mail: crissonicx2@gmail.com
‡e-mail: vidarr.vali@gmail.com
§e-mail: jpssousa97@gmail.com
¶e-mail: vitormd96@gmail.com

2.1 Fundamentação Teórica

A base teórica do projeto consiste em alguns tópicos do livro do Patterson [1]. Mais precisamente: o Assembly RISC-V, o processador multiciclo, seu *datapath*, o verilog utilizado para desenvolvê-lo e a memória *cache*.

2.1.1 Assembly RISC-V

A programação em Assembly na ISA RISC-V é bem similar à arquitetura MIPS, com algumas diferenças em relação a instruções de multiplicação e nomenclatura de registradores. Um exemplo claro é o uso da instrução de multiplicação: onde no MIPS a instrução guarda o resultado em dois registradores (um para a parte superior, mais significativa e outro para a parte inferior, menos significativa), o RISC-V guarda o resultado da multiplicação direto no registrador de destino com a instrução determinando qual metade do número resultante será guardado (mul para guardar a parte menos significativa, mulh para a parte mais significativa). Os caminhos de dados das duas arquiteturas também são diferentes.

A ISA RISC-V utilizada no projeto suporta valores inteiros e instruções de multiplicação. Não faz nenhum uso de valores em ponto flutuante (*float*).

2.1.2 Processador Multiciclo

Durante o semestre, os alunos da disciplina de OAC, desenvolveram os processadores uniciclo e multiciclo para a parte prática da disciplina. O projeto de desenvolvimento do jogo deveria utilizar um dos processadores desenvolvidos, à escolha dos alunos.

Entre as duas opções, o processador melhor implementado, pelos autores desse artigo, foi o multiciclo e, portanto, foi o escolhido para o desenvolvimento.

O processador multiciclo é baseado em uma ideia relativamente simples: cada instrução é executada em múltiplos ciclos de *clock*. Seu caminho de dados (*datapath*) pode ser observado na Figura 2. O caminho de dados de um processador é o conjunto de circuitos digitais que define a rota que uma instrução faz para ser executada corretamente, no que diz respeito à sua definição perante a arquitetura do processador.

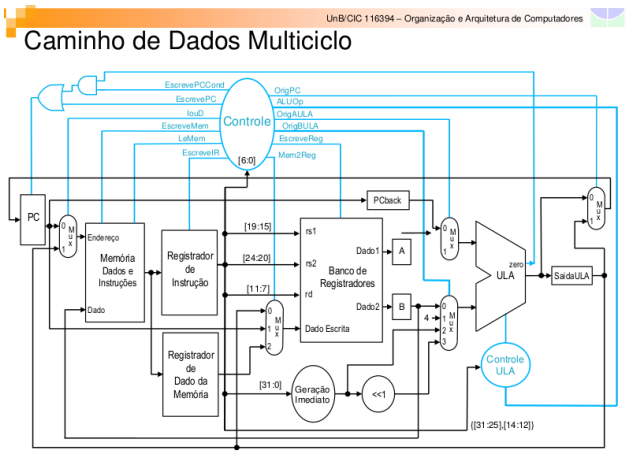

Figura 2: Caminho de dados do Multiciclo.

2.1.3 System Verilog

Para o desenvolvimento dos processadores, os alunos utilizaram o Verilog, uma linguagem de descrição de hardware. Em conjunto com o Quartus - software para o desenvolvimento, análise e sintese de projetos Verilog - o código foi compilado para gerar um arquivo capaz de ser executado na DE1-SoC e tornar a utilização real do processador desenvolvido possível.

2.1.4 Memória Cache

A memória cache é a de maior velocidade - e maior preço - em um computador. Faz parte do processador e pode possuir diferentes níveis (L1, L2...), cada nível um pouco mais lento que o anterior e com mais memória disponível.

Para os processadores desenvolvidos na disciplina, os alunos só teriam acesso a memória cache - o que representa uma limitação considerável, uma vez que o tamanho final do código deve ser menor ou igual ao tamanho da cache - de 16KiB para instruções e 8KiB para dados.

Para o projeto aqui apresentado, inicialmente foram desenvolvidas duas telas para o jogo porém, as imagens ocupavam muito espaço da memória e tornaram-se impossíveis de serem utilizadas na DE1-SoC. Foi necessário pensar uma estratégia que possibilitasse a impressão do tabuleiro e das peças ocupando uma porção menor na memória. A estratégia utilizada pode ser verificada na seção 2.2.3.

2.2 Fundamentação Técnica

A base técnica do projeto consiste em assuntos mais específicos, postos em prática a partir da fundamentação teórica: as falhas encontradas na programação Assembly, o assembler RARS, a limitação e estratégia de uso da memória, o software Quartus e o hardware FPGA da Altera

2.2.1 Programação em Assembly

O projeto foi dividido em alguns módulos para facilitar o desenvolvimento e divisão de tarefas entre os membros da equipe. Entre eles, o módulo gráfico (que escreve dados diretamente na memória de vídeo, possibilitando a impressão do tabuleiro e das peças e suas movimentações na tela), o módulo sonoro (que escreve dados diretamente na memória de som, utilizando métodos que simulavam *ecall*s de saídas MIDI), o módulo de entrada (que decodifica qual botão do teclado foi pressionado) e o módulo lógico (módulo principal do jogo, que realiza todas as regras de um jogo de damas: desde a movimentação de uma peça até a verificação de uma condição de vitória ou derrota e uma inteligência artificial que joga contra o usuário).

O módulo gráfico é bastante simples: ao receber parâmetros de qual tipo de peça e localização, imprime a imagem desejada no tabuleiro, na posição desejada. Além disso, possui uma função para imprimir o tabuleiro e o menu de seleção de dificuldade na tela.

O módulo sonoro recebe um parâmetro (qual som tocar) e executa uma sequência de sons dependendo desse parâmetro. Os sons que podem ser executados são os seguintes: movimentação, captura, promoção de peças, fanfarra de vitória, fanfarra de derrota, som de movimentação inválida.

O módulo de entrada recebe as teclas pressionadas pelo teclado e as decodifica para enviar ao módulo lógico, que irá tratá-las. É um módulo padrão que está presente em muitos jogos.

O módulo lógico é, com certeza, o maior e mais importante módulo do projeto. Todas as regras, jogadas e peculiaridades do jogo de damas estão implementadas aqui. Ele executa uma série de laços e verificações, em conjunto com o módulo de entrada, para verificar se o jogador solicitou uma jogada válida, se uma peça comeu outra, se uma peça chegou ao outro lado do tabuleiro para ser promovida e se ainda há peças do lado do jogador ou do computador.

2.2.2 RARS

O RARS é uma ferramenta para simular execuções de Assembly RISC-V e montar (realizar o trabalho de um *assembler*) programas em linguagem de máquina. Foi utilizado pela equipe para o desenvolvimento, *debugging* e, após a conclusão, exportar os dados da memória de instrução e dados para serem executados no processor - arquivos .mif que são usados na simulação com a placa FPGA.

A ferramenta dispõe de duas abas: uma para edição de texto, que representa o código que será simulado e outra que mostra todas as informações sobre endereços de instruções, valores dos registradores e valores armazenados na memória, além de um pequeno espaço para receber os retornos do programa (se houve algum erro, se o programa foi interrompido, etc.).

Pelo fato do RARS não ser uma ferramenta plenamente finalizada, apresentou muitos erros e falhas ao testar o jogo nele. Apesar disso, a equipe se esforçou o máximo para evitar que quaisquer problemas da própria ferramenta interfiram no funcionamento do projeto.

2.2.3 A memória

A memória está sendo utilizada, principalmente, para armazenar os endereços de retorno a instruções, devido ao alto número de pulos incondicionais e condicionais (*jal*s e *branch*es) que o código necessita para executar todas as suas funcionalidades; se os endereços de retorno não forem guardados, o programa não executará corretamente e acessará instruções que não condizem com a lógica do jogo. Além disso, a memória também serve para guardar valores que estão armazenados em registradores de função (registradores *a0* até *a7*) na mudança de escopo, para que não sejam perdidos ou modificados erroneamente.

Como dito anteriormente, a memória disponível para a execução do jogo na FPGA é bastante limitada (16KiB para instruções e 8KiB para dados). Inicialmente, foram desenvolvidas duas telas iniciais e mais 5 imagens para realizar os movimentos do jogo. As telas iniciais podem ser vistas abaixo:

Porém, as imagens necessitavam muito espaço na memória - espaço que não estava disponível - e, portanto, foi necessário uma estratégia para contornar essa limitação. A estratégia pensada foi: a retirada do menu inicial e do tabuleiro inicial. Depois, foi desenvolvida uma pequena imagem para cada *tile* do jogo que foi replicada 32 vezes, formando o tabuleiro inicial. Após o tabuleiro ser gerado, a aplicação iria imprimir as peças do jogo. A mudanças de estratégia gerou uma mudança significativa. O tamanho do arquivo

Figura 3: Menu inicial.

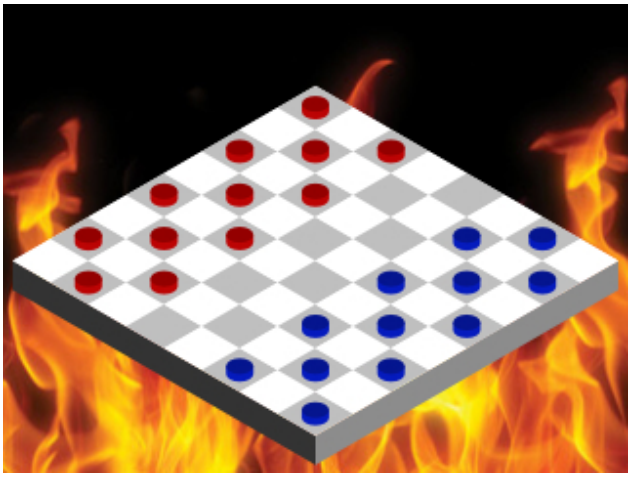

Figura 4: Tabuleiro inicial.

escrito no RARS diminuiu de 14807 linhas (978,6kB) para 2359 linhas (52,2kB), uma redução para 5,33% do tamanho original.

2.2.4 Quartus e FPGA

O Quartus é um software de muitos usos para a programação em hardware. A equipe o utilizou ao longo do semestre para o desenvolvimento dos processadores mencionados na Introdução e realizar os testes no dispositivo FPGA DE1-SoC (Figura 5).

Como dito na seção de fundamentos teóricos, o grupo optou por utilizar seu processador multiciclo. Dito isto, o processador não sofreu modificações desde sua entrega no Laboratório 4. A diferença foi qual código em Assembly ele recebeu na função *Programmer* do Quartus.

Para que o grupo pudesse testar seu projeto na placa DE1-SoC, foi necessário compilar o código no RARS, solicitar um *Dump Memory* e carregar os arquivos ".mif"gerados na mesma função *Programmer* do Quartus. No momento que fosse iniciado o processador com essas memórias de dados e instruções, o jogo se iniciava e o grupo pode testá-lo, até que todos os bugs fossem corrigidos.

3 METODOLOGIA

A metodologia utilizada para desenvolver o projeto envolveu diversos métodos ensinados em outras disciplinas do curso, como Engenharia de Software. Entre elas estão o *scrum*, uso de controle de versão e rotinas de teste de módulos.

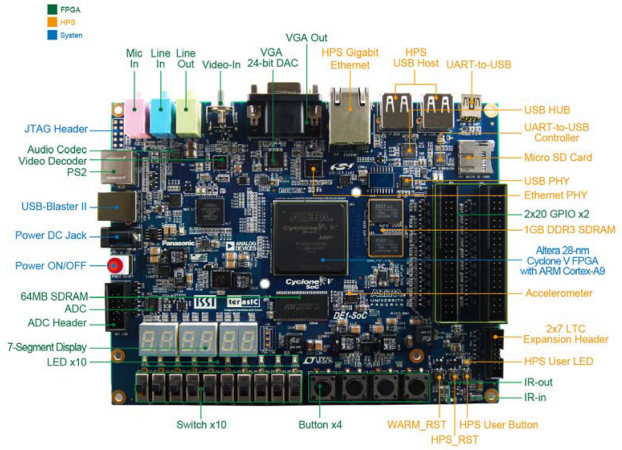

Figura 5: Diagrama da placa DE1-SoC.

Como o projeto envolvia o uso de diversas tecnologias do RARS e da DE1-SoC, optamos por dividir a equipe em 3 sub-grupos:

1. Desenvolver a lógica do jogo: carregar o tabuleiro, avaliar o input do usuário, gerar o input da CPU (dado o nível de dificuldade) e realizar as mudanças de acordo com as regras do jogo.

2. Construir o módulo de gráfico e audio levando em conta as tecnologias utilizadas pelo RARS e a placa DE1-SOC

3. Avaliar o funcionamento do processador RISC-V, e em conjunto com os outros grupos, realizar o *port* do jogo para a FPGA levando em conta as limitações apresentadas do projeto (ex: tamanho da memória de dados).

Ao decorrer do projeto, cada grupo deveria documentar as soluções e dificuldades encontradas com intuito de contribuir para o relatório final do projeto.

Nas últimas 3 (três) semanas do curso, utilizamos o scrum para verificar o progresso realizado em cada equipe. Na 1ª semana apresentamos um avanço significativo na etapa mais importante do projeto: conjunto de soluções em papel para fazer o jogo funcionar. Esta consiste em, por exemplo, como será impresso o tabuleiro na tela, quais regras do jogo original de damas serão adotadas, quais são os recursos utilizávies em cada uma das plataformas-alvo.

A 2ª semana foi a de implementação e realização de rotinas de teste. Estas rotinas são de suma importância para avaliar se mudanças futuras não afetarão o funcionamento correto do jogo. Isto foi útil para a confecção da versão do jogo para a placa DE1-SOC.

A última semana consistiu em realizar testes com a versão para o RARS a fim de encontrar *bugs*, a adaptação do jogo para a placa e confecção do relatório a partir da documentação interna realizada nas últimas semanas.

Vale ressaltar que para construir os módulos do jogo (2ª semana), utilizamos a abordagem predominantemente *bottom-up*, ao contrário da 1ª semana. Isto porque era necessário verificar aos poucos se os componentes estavam funcionando como o esperado.

Com relação ao controle de versão, adotamos o software git em conjunto com o GitHub para compartilhar as modificações no código do projeto ao longo do tempo. Esse foi de grande utilizade para verificar o que cada integrante da equipe fez, e caso tudo esteja

conforme o combinado, aplicar as alterações no produto final, de modo que não haja surpresas desagradáveis durante a apresentação.

Tendo em vista o resultado, é notório o êxito na escolha das ferramentas e estratégias adotadas durente o desenvolvimento do projeto. Concluímos também que não basta utilizar uma única metodologia porque etapas distintas podem demandar outras abordagens para conclui-las com eficácia.

4 Resultados Obtidos

Ao final do desenvolvimento, obtemos duas versões para o jogo de DAMAS: uma mais completa para ser executada no simulador RARS, e outra com alguns recursos a menos voltada para a FPGA DE1-SoC. Estas limitações (descritas nas seções 2.1.4. e 2.2.3.) não afetaram a jogabilidade proposta no modelo teórico. Ressaltamos que este tipo de situação é comum quando se trata de sistemas mais especializados: na década de 90, consoles de mesa tinham capacidades de processamento decisórias, portanto múltiplas versões do mesmo jogo eram lançadas. Neste capítulo será apresentada o funcionamento de cada versão:

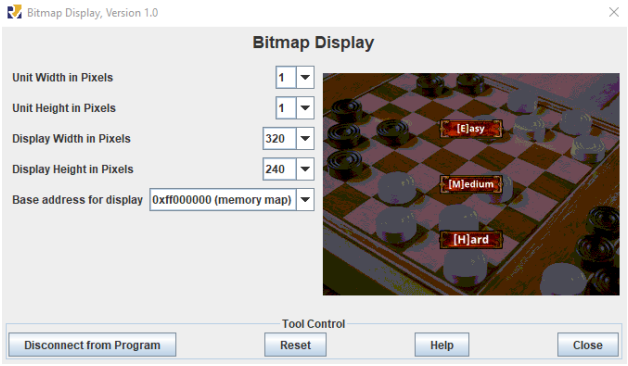

Figura 6: Menu do Jogo

A Figura 6 apresenta o funcionamento do jogo na plataforma RARS.

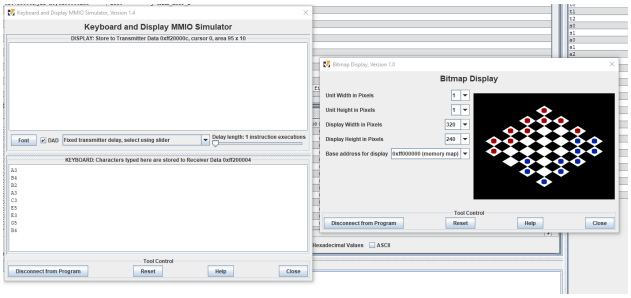

Figura 7: Versão da DE1-SoC

A Figura 7 apresenta o funcionamento do jogo na placa DE1-SoC.

5 Conclusão

O projeto foi uma oportunidade de colocar em prática os conceitos vistos em sala na disciplina de Organização e Arquitetura de Computadores. A ISA RISC-V ainda é bastante nova e, tendo em vista as dificuldades encontradas na utilização do RARS, fica claro que ainda necessita de desenvolvimento e pesquisa. No entanto, é possível verificar que a ISA é passível de ser utilizada como ferramenta educacional para alunos do curso de Ciência da Computação e áreas correlatas.

Referências

[1] J. L. H. David A. Patterson. *Computer Organization and Design: The Hardware Software Interface [RISC-V Edition]*. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 1st edition, 2017.