UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE CICLOS DE INSTRUÇÕES

APUCARANA – PR
2023

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE CICLOS DE INSTRUÇÕES

Trabalho apresentado à disciplina de Arquitetura e Organização de Computadores, do curso de Bacharelado em Ciência da Computação.

Professor: Guilherme Nakahata

APUCARANA – PR
2023

SUMÁRIO

INTRODUÇÃO

No domínio dinâmico da ciência da computação e do estudo da arquitetura de computadores, uma compreensão dos intrincados processos que ocorrem dentro de uma Central Processing Unit (CPU) é fundamental. Um dos inúmeros aspectos desta compreensão é o conceito de ciclos de instrução, que são fundamentais para governar a execução de instruções dentro de uma CPU. Este trabalho oferece uma exploração interativa e prática do processamento de instruções da CPU e da função dos ciclos de instrução.

Os ciclos de instrução são as batidas rítmicas que orquestram a operação de uma CPU, ditando as etapas sequenciais envolvidas na busca, decodificação, execução e armazenamento de instruções. No contexto mais amplo da arquitetura de computadores, os ciclos de instrução são o eixo que conecta o mundo abstrato das linguagens de programação à realidade tangível das operações da CPU. Eles preenchem a lacuna entre o código de alto nível e o hardware de baixo nível, facilitando a execução de instruções.

Este trabalho, embora focado no conceito crítico de ciclos de instrução, transcende o domínio teórico. Ele foi projetado para capacitar os alunos a se envolverem em um aprendizado prático, desenvolvendo um programa simulador de fila de instruções de CPU. Isso fornece uma plataforma tangível para os alunos explorarem a operação de filas de instruções, endereços de memória e registros de buffer de memória (MBRs) no contexto de uma CPU.

Nas seções a seguir, os detalhes da implementação desta aplicação serão explorados, que trata-se no desenvolvimento de uma simulação abrangente e fácil de usar da fila de instruções de uma CPU. Através das lentes desta simulação, será possível obter uma experiência prática dos conceitos de arquitetura de CPU, algoritmos e engenharia de software, melhorando, em última análise, a compreensão do papel vital desempenhado pelos ciclos de instrução na operação de dispositivos de computação modernos.

CAPÍTULO 1

MOTIVAÇÃO E RECURSOS UTILIZADOS DURANTE A IMPLEMENTAÇÃO

Na jornada de desenvolvimento de software, a implementação bem-sucedida de um projeto depende da seleção e utilização criteriosa de recursos. Estes recursos abrangem uma ampla gama de elementos, desde linguagens de programação e bibliotecas até estruturas de dados, ferramentas e metodologias. Neste capítulo, será aprofundado na intrincada rede de recursos que desempenharam um papel fundamental na definição do resultado deste projeto.

Ao dissecar esses componentes e explorar a lógica por trás de sua seleção e utilização, será esclarecido como a aplicação foi elaborada e quais escolhas foram feitas em cada etapa.

1.1 LINGUAGENS DE PROGRAMAÇÃO E BIBLIOTECAS

Inicialmente, a lógica central da aplicação foi desenvolvida em Java, principalmente por ser a linguagem de programação enfatizada nas aulas universitárias do autor. Java é conhecido por sua portabilidade e robustez, tornando-o uma escolha popular para instituições educacionais e desenvolvimento de software. No entanto, à medida que o projeto avançava, tornou-se evidente que a biblioteca gráfica JavaFX, comumente usada com Java para criar interfaces de usuário, apresentava alguns desafios. A experiência do autor com JavaFX não foi ideal e ficou evidente que era necessário explorar opções alternativas para melhorar a interface do usuário e a experiência geral de desenvolvimento.

Na busca por uma alternativa, Python surgiu como uma escolha atraente. Python é conhecido por sua simplicidade e legibilidade, o que o torna uma linguagem excelente para o rápido desenvolvimento de aplicativos. Embora possa não ter o mesmo desempenho que o Java em termos de velocidade de execução bruta, ele se destaca por fornecer um ambiente de desenvolvimento mais rápido, permitindo que os desenvolvedores iterem rapidamente em seus projetos.

A verdadeira virada de jogo, entretanto, foi a descoberta do PyQt6, uma ligação Python para o framework Qt. As capacidades do Qt na criação de interfaces gráficas multiplataforma e sua extensa biblioteca de widgets e ferramentas abriram novas possibilidades para o projeto. A integração do PyQt6 ao projeto possibilitou a utilização do poder da estrutura Qt e, ao mesmo tempo, a sintaxe concisa e expressiva do Python. Essa combinação não apenas melhorou a qualidade da interface gráfica, mas também acelerou significativamente o processo de desenvolvimento. O resultado foi um aplicativo mais sofisticado e fácil de usar, que se beneficia do melhor dos dois mundos: a versatilidade do Qt e a agilidade do Python.

A título de esclarecimento, Qt é um *framework* popular e robusto para a criação de interfaces gráficas, que possui uma comunidade de usuários grande e ativa, isso significa que os desenvolvedores têm acesso a uma grande variedade de recursos, incluindo fóruns, listas de discussão e documentação online. Este apoio comunitário pode ser inestimável quando se deparam com desafios durante o desenvolvimento.

Além disso, Qt é conhecido por seu desempenho, tornando-o adequado para aplicações que exigem capacidade de resposta em tempo real e gerenciamento eficiente de recursos. Isso é especialmente crucial para aplicativos como jogos, simulações e software de desktop de alto desempenho.

A arquitetura modular do Qt permite que os desenvolvedores selecionem apenas os componentes necessários, minimizando o tamanho e a complexidade da aplicação. Além disso, o Qt pode ser facilmente estendido com código personalizado, garantindo flexibilidade e adaptabilidade para requisitos específicos do projeto.

1.2 FERRAMENTAS DE BUILD

Outrossim, o projeto utiliza o pacote PyInstaller, o que permite aos desenvolvedores empacotar aplicativos Python em executáveis independentes, facilitando a distribuição de software aos usuários finais. Isso elimina a necessidade dos usuários instalarem o Python e as dependências separadamente, simplificando o processo de instalação.

O pacote agrupa automaticamente o interpretador Python e as dependências necessárias no executável, eliminando a necessidade de gerenciamento manual de dependências. Isso reduz problemas de compatibilidade e torna a implantação mais confiável.

Ademais, é uma ferramenta de código aberto, o que significa que está disponível gratuitamente e pode ser usada para projetos pessoais e comerciais sem custos de licenciamento. Isto o torna uma escolha atraente para projetos com restrições orçamentárias.

Por outro lado, aplicativos Java empacotados como arquivos .jar podem encontrar problemas de execução, muitas vezes exigindo que os usuários naveguem na linha de comando ou criem arquivos .bat, o que pode ser confuso para usuários não técnicos.

Ora, aplicativos Java geralmente exigem que os usuários tenham a versão correta do Java Runtime Environment (JRE) instalada, levando a incompatibilidades de versão e erros de tempo de execução.

Por fim, para atingir o objetivo de multiplataforma, tendo em vista que o ambiente de desenvolvimento do autor é o Windows 10 Enterprise LTSC, foi utilizada uma combinação de WSL (Windows Subsystem for Linux), Oracle VM VirtualBox e Ubuntu.

O WSL permite que os desenvolvedores executem uma distribuição Linux junto com o Windows, fornecendo uma maneira perfeita de desenvolver e testar aplicativos compatíveis com Linux em uma máquina Windows. Essa compatibilidade entre plataformas garante que os desenvolvedores possam trabalhar em seu ambiente Windows preferido enquanto direcionam executáveis Linux.

Todavia, como trata-se de uma aplicação com interface gráfica, a Oracle VM VirtualBox também foi utilizada, uma plataforma de virtualização que permite a criação de máquinas virtuais (VMs) isoladas. Ao configurar uma VM Ubuntu no VirtualBox, os desenvolvedores podem criar um ambiente de teste Linux dedicado, garantindo que o executável gerado pelo PyInstaller funcione corretamente em um sistema Linux genuíno.

O Ubuntu é uma das distribuições Linux mais populares, com ampla compatibilidade e uma grande base de usuários. Desenvolver e testar no

Ubuntu garante compatibilidade com uma parte significativa da comunidade de usuários Linux.

Depois de testar e empacotar o aplicativo Python no Ubuntu, o executável compatível com Linux resultante pode ser distribuído aos usuários Linux com confiança, sabendo que foi testado em um ambiente Linux autêntico.

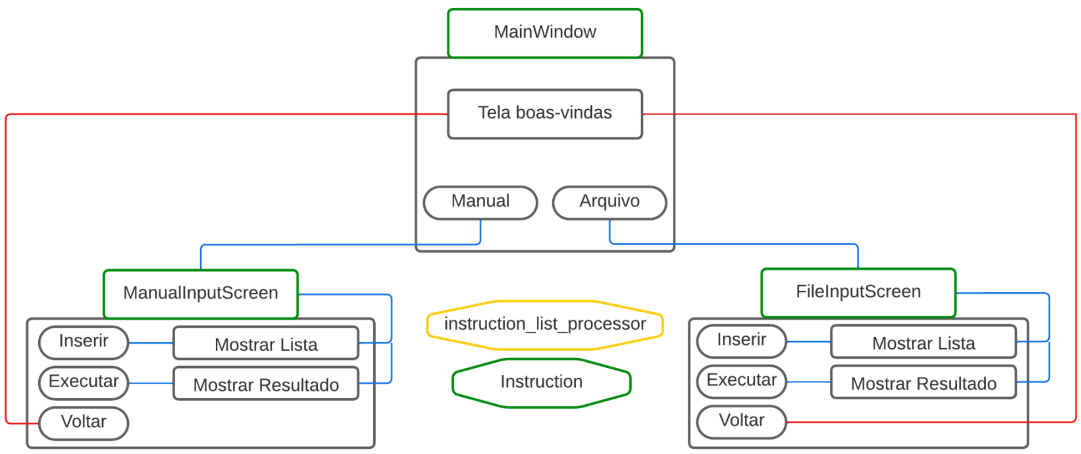
1.3 ESTRUTURA DA INTERFACE GRÁFICA

Neste projeto, uma abordagem de layout empilhado é adotada para gerenciar diferentes telas ou layouts de forma eficiente. A aplicação inicia com uma tela de boas-vindas e, à medida que o usuário interage com a GUI, novos layouts são carregados, adicionados a uma pilha e exibidos com base em suas escolhas.

Essa abordagem oferece vários benefícios como a modularidade, onde cada tela ou layout é desenvolvido de forma independente, facilitando o foco em um aspecto do aplicativo por vez.

Ademais, tem-se uma melhor eficiência de memória, pois a medida que o usuário avança, novos layouts são adicionados à pilha. No entanto, se o usuário optar por voltar, o layout anterior será removido da pilha e efetivamente “destruído”. Isso ajuda a economizar memória e recursos, garantindo que o aplicativo permaneça responsivo e eficiente mesmo durante fluxos de trabalho complexos.

Finalmente, temos uma separação relativa de funções, onde cada layout pode encapsular sua própria lógica e funcionalidade, promovendo uma divisão mais clara de interesses na base de código do aplicativo.



1.4 ESTRUTURAS DE DADOS PARA A LÓGICA DA APLICAÇÃO

Neste projeto, a estrutura de dados que desempenhou o papel fundamental no gerenciamento da simulação foi a array, ou list no Python. Duas dessas matrizes foram usadas para armazenar linhas de instrução e registrar eventos, oferecendo uma maneira simples, porém eficaz, de organizar e acompanhar o progresso da simulação.

A primeira matriz foi empregada para armazenar linhas de instrução, cada linha consistindo em um código de instrução e um endereço de memória. Essa matriz serviu como estrutura de dados primária para imitar a fila de instruções em um processador. Permitiu o armazenamento e recuperação eficiente de instruções, facilitando seu processamento durante a simulação.

A segunda matriz foi dedicada a registrar eventos e ações que ocorreram durante cada etapa de execução da simulação. O log é um aspecto crucial de qualquer simulação, pois fornece informações valiosas sobre o comportamento do sistema. Essa matriz foi fundamental no registro de informações sobre a execução de cada código de instrução, o que eles fizeram e como o MBR foi modificado.

Para executar a sequência de instruções e controlar o fluxo da simulação, um loop while encapsula a lógica central do projeto. Dentro deste loop, um switch case foi empregado para determinar qual código de instrução seria executado para cada linha. A estrutura switch case é particularmente adequada para cenários onde múltiplas ramificações de execução precisam ser tratadas com base no valor de uma variável específica (neste caso, o código de instrução). À medida que o loop iterava pelas linhas de instrução, o switch case direciona eficientemente o programa para o bloco de código apropriado para execução. O loop continua até que toda a sequência de instruções é processada, ou algum gatilho de erro é acionado. Esta abordagem garantiu que a simulação executasse as instruções de forma controlada e organizada.

Em resumo, o uso de arrays para armazenar linhas de instrução e registrar eventos, combinado com uma estrutura de switch case encapsulada em um loop while, formou a espinha dorsal da lógica central da simulação. Essas estruturas de dados e mecanismos de fluxo de controle simples, porém eficazes, permitiram a representação precisa de uma fila de instruções e a execução controlada da sequência de instruções.

Para agilizar o gerenciamento de instruções dentro da simulação, foi introduzida uma classe "Instruction" simples, mas versátil. Esta classe foi projetada para representar instruções individuais a serem armazenadas nos arrays da simulação. A classe "Instruction" encapsula os principais atributos que definem uma instrução no contexto da simulação: "code", "value_a" e "value_b". Esses atributos permitem coletivamente a representação flexível de simples gama de instruções da arquitetura fictícia estipulada nos requisitos do projeto.

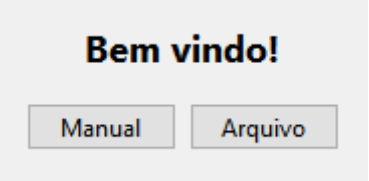
O atributo "code" serve como o identificador de cada instrução, capturando sua operação ou função principal. Este código é fundamental para a execução e simulação de instruções dentro do processador, pois orienta a lógica da simulação.

Os atributos “value_a” e “value_b” são opcionais, reconhecendo que nem todas as instruções requerem dados adicionais além de sua operação principal. Para muitas instruções, apenas o “code” e o “value_a” são suficientes para definir sua funcionalidade. No entanto, certas instruções, como aquela com o código “000010”, dependem de “value_a” e “value_b” para especificar sua operação com precisão. Esses atributos opcionais permitem a representação de diferentes formatos de instrução, garantindo que a simulação possa imitar com precisão o comportamento de diversas instruções do processador.

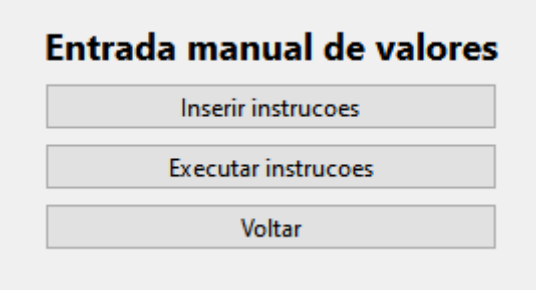
CAPÍTULO 2

RESULTADOS

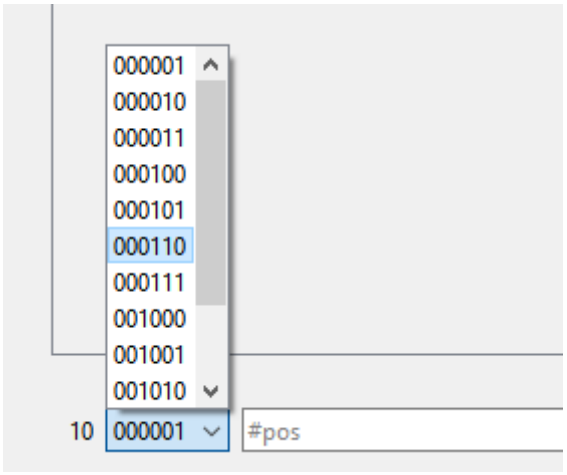
A aplicação oferece uma plataforma interativa e fácil de usar para simular a fila de instruções de uma CPU, permitindo aos usuários explorar e experimentar os conceitos fundamentais do processamento de instruções. O aplicativo é organizado em dois modos principais de entrada: “Manual” e “Arquivo”, cada um adaptado para acomodar diferentes preferências e cenários do usuário.



No modo de entrada “Manual”, os usuários recebem três botões essenciais: “Inserir”, “Executar” e “Voltar”.

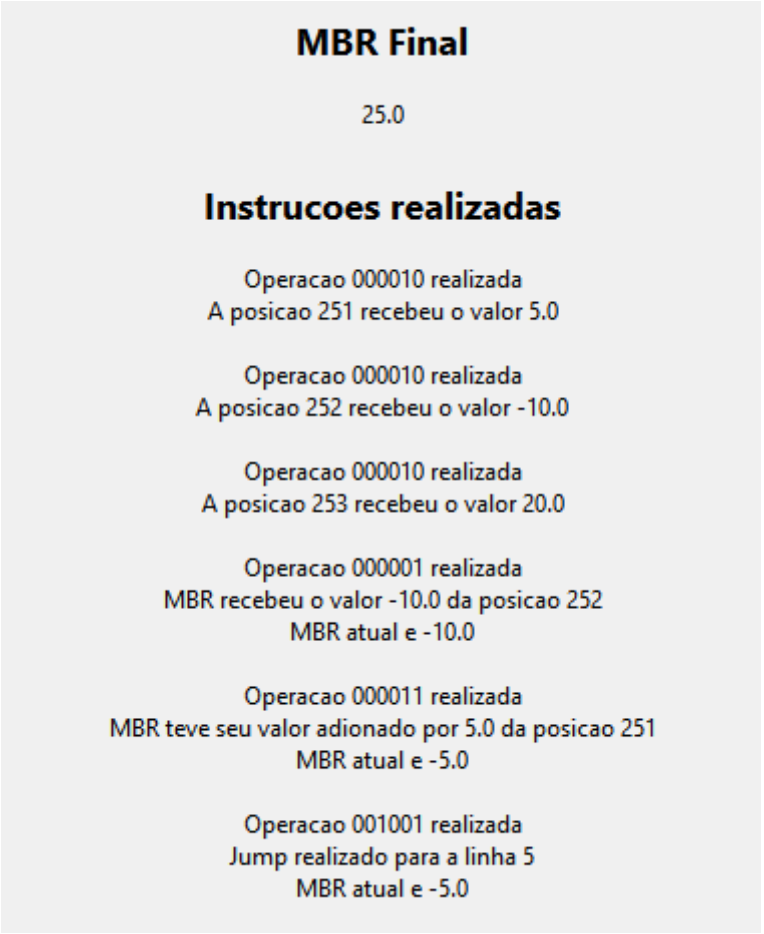


Clicar no botão "Inserir" leva os usuários a uma tela abrangente onde eles podem criar e adicionar instruções interativamente à simulação. A experiência do usuário é aprimorada com um seletor que contém uma lista de todos os códigos de instrução disponíveis, garantindo precisão e facilidade de uso. Dependendo da instrução selecionada, os campos de entrada tornam-se ativos ou inativos para solicitar aos usuários dados específicos, como endereços de memória ou valores. Os usuários podem adicionar essas instruções a uma fila visual, onde podem observar a sequência de instruções e até mesmo remover instruções específicas para ajustar sua simulação.



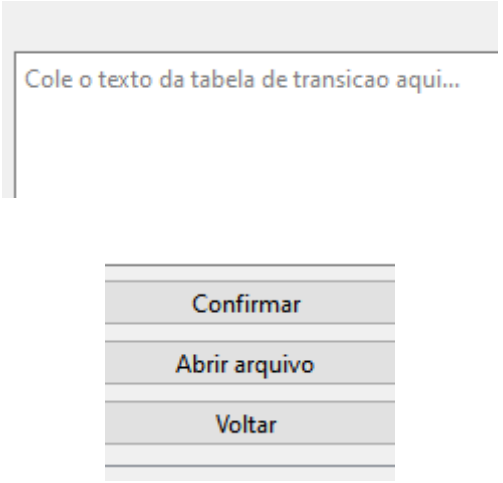
Instrucao	Codigo da Instrucao	Operandos	Remover?
1	000010	#pos 251 #dado 5.0	Remover
2	000010	#pos 252 #dado -10.0	Remover
3	000010	#pos 253 #dado 20.0	Remover
4	000001	#pos 252	Remover
5	000011	#pos 251	Remover
6	001001	#lin 5	Remover
7	001000	#lin 5	Remover
8	000011	#pos 253	Remover
9	001100		Remover

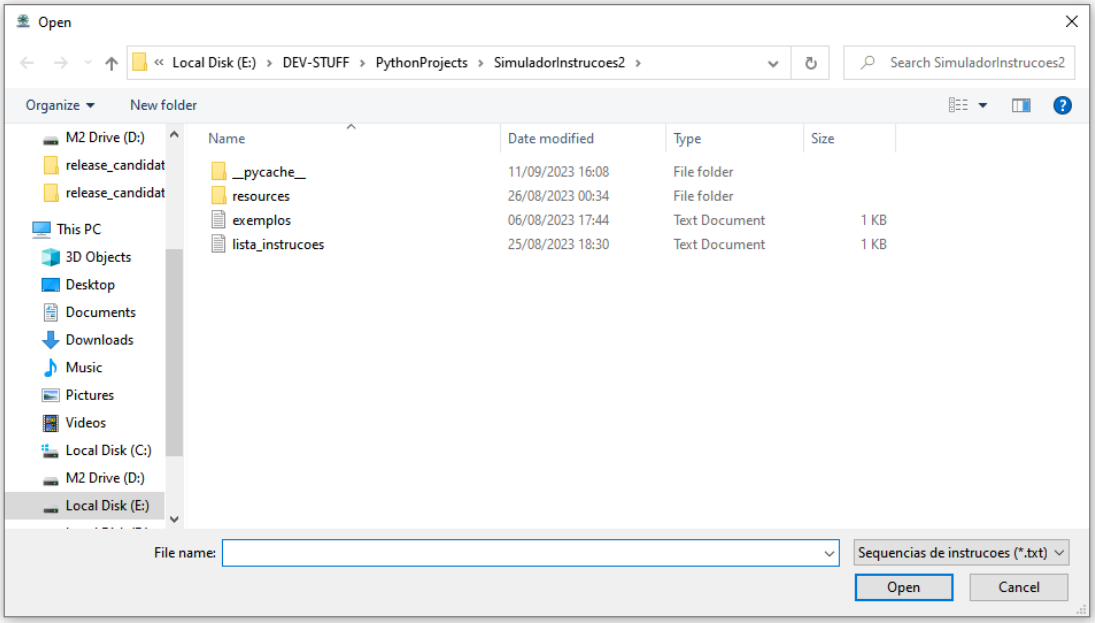
O botão “Executar” guia os usuários para uma tela dedicada que exibe um registro de todas as instruções executadas e seus resultados. Este log fornece informações sobre as alterações nos endereços de memória e os valores resultantes do Memory Buffer Register (MBR). Os usuários podem usar esse log para acompanhar o progresso de sua simulação e obter uma compreensão mais profunda de como as instruções são processadas em uma CPU.



A qualquer momento no modo de entrada “Manual”, os usuários podem retornar à tela de boas-vindas clicando no botão “Voltar”, garantindo uma experiência de navegação intuitiva e contínua.

Alternativamente, o modo de entrada “Arquivo” simplifica o processo de fornecimento de instruções à simulação, atendendo aos usuários que preferem trabalhar com vários conjuntos de instruções ou conjuntos de dados maiores.





Neste modo, os usuários têm a opção de colar um texto formatado representando códigos de instrução em uma caixa de texto ou usar um conveniente botão "Abrir Arquivo" para abrir um navegador de arquivos para abrir .txts de entrada semelhante. Essa flexibilidade permite que os usuários escolham o método de entrada que atenda às suas necessidades.

Neste modo, a aplicação aceita conjuntos de instruções em formato simples e estruturado. Cada instrução é representada da seguinte forma: "[CODE],[VALUE_A],[VALUE_B]" em uma linha separada, dividindo cada valor com uma vírgula. O formato garante clareza e flexibilidade, acomodando diversas instruções que podem exigir um, dois ou nenhum valor, dependendo do código específico utilizado.

Após fornecer o conjunto de instruções, o usuário pode executar a simulação clicando no botão “Executar”. O aplicativo processa as instruções fornecidas e exibe resultados semelhantes ao modo de entrada "Manual", incluindo um log de instruções executadas, alterações nos endereços de memória e valores MBR.

Da mesma forma como acontece com o modo de entrada “Manual”, os usuários podem retornar à tela de boas-vindas clicando no botão “Voltar”, garantindo uma interface consistente e amigável.

A aplicação oferece uma experiência envolvente e educacional para usuários interessados em explorar o funcionamento interno do processamento

de instruções da CPU. A interface intuitiva, a criação dinâmica de instruções e os logs detalhados de resultados permitem que os usuários experimentem, aprendam e obtenham insights práticos sobre a operação de uma fila de instruções dentro de uma CPU.

Seja usando a entrada "Manual" ou "Arquivo", os usuários podem interagir facilmente com o programa, tornando-o uma ferramenta valiosa tanto para iniciantes quanto para aqueles que desejam aprofundar sua compreensão da arquitetura do processador e do processamento de instruções.

CONCLUSÃO

O desenvolvimento desta aplicação atingiu seus objetivos educacionais, bem como demonstrou o poder das modernas ferramentas e metodologias de desenvolvimento de software. Através desta tarefa, os conteúdos teóricos apresentados na sala de aula foram aprofundados de maneira mais prática com a simulação da fila de instruções de uma CPU.

Ao criar esta aplicação, os recursos do framework Qt com o pacote PyQt6, permitiu casar a agilidade do Python com a robustez das proezas gráficas do Qt. Essa escolha foi fundamental para atingir os objetivos de forma eficiente, já que a simplicidade do Python e o ambiente rico em widgets do PyQt6 facilitaram o desenvolvimento rápido sem comprometer a funcionalidade ou a facilidade de uso do aplicativo.

Além disso, o pacote PyInstaller em conjunto com recursos WSL do Windows permitem a build e distribuição da aplicação para outras plataformas como o Linux bem como a utilização do arquivo através de um executável ou arquivo binário.

A inclusão de dois modos de entrada distintos, “Manual” e “Arquivo”, proporciona mais flexibilidade ao usuário. O modo "Manual" permite que os usuários criem e visualizem sequências de instruções de forma interativa, promovendo uma compreensão profunda das filas de instruções da CPU. Enquanto isso, o modo “Arquivo” agiliza o processo para testar rapidamente diversos conjuntos de instruções predefinidos, garantindo que o aplicativo atenda a uma ampla gama de preferências do usuário.

Concluindo, a atribuição simulada de fila de instruções de CPU não serviu apenas para o estudo da disciplina de Arquitetura e Organização de Computadores, mas também como uma prática multidisciplinar de desenvolvimento de software, design de GUI e interação do usuário.

REFERÊNCIAS

Java. Disponível em: https://www.java.com. Acesso em: 23 jul. 2023.

OpenJFX. Disponível em: https://openjfx.io/. Acesso em: 23 jul. 2023.

Python. Disponível em: https://www.python.org/. Acesso em: 23 jul. 2023.

Qt. Disponível em: https://www.qt.io/. Acesso em: 23 jul. 2023.

PyQt6 - PyPI. Disponível em: https://pypi.org/project/PyQt6/. Acesso em: 23 jul. 2023.

PyQt - Riverbank Computing. Disponível em: https://www.riverbankcomputing.com/software/pyqt/. Acesso em: 23 jul. 2023.

PyInstaller - PyPI. Disponível em: https://pypi.org/project/pyinstaller/. Acesso em: 23 jul. 2023.

PyInstaller Documentation. Disponível em: https://pyinstaller.org/en/stable/. Acesso em: 23 jul. 2023.

Windows Subsystem for Linux Installation Guide. Disponível em: https://learn.microsoft.com/en-us/windows/wsl/install. Acesso em: 23 jul. 2023.

Ubuntu. Disponível em: https://ubuntu.com/download. Acesso em: 23 jul. 2023.

VirtualBox. Disponível em: https://www.virtualbox.org/. Acesso em: 23 jul. 2023.