Grace: um simulador de MIPS para estudo da Arquitetura de Computadores

Luan Silva de Sena Advincula

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

lssa@cin.ufpe.br

Abstract. *In the Hardware Infrastructure course, essential components for computer operation are covered, including processing units, memory, and storage. The interface between hardware and software is provided by the Instruction Set Architecture (ISA), commonly known as the instruction set in Portuguese, using an assembly language implementation generated by high-level language compilers. This language uses mnemonics that abstract the binary representation of instructions and data, making CPU programming easier and more intuitive. In the course, students develop Assembly code to understand how instructions are executed by the processor and how compilers generate this language from higher-level code. Tools such as instruction set simulators are used to read the code and generate outputs based on it, displaying data in registers for correctness verification. These tools generally include their own text editor and other graphical elements to guide the user. Since there are few similar tools, many of them are not cross-platform and are restricted to specific operating systems, an open-source, educational-purpose Assembly interpreter in a command-line format was developed. The goal is to make it cross-platform and easy to use without the need for a specific text editor. This will facilitate the development of activities in the Hardware Infrastructure course and will be available for free to any interested user.*

Resumo. *Na disciplina de Infraestrutura de Hardware, são abordados os componentes essenciais para o funcionamento do computador, incluindo unidades de processamento, memória e armazenamento. A interface entre hardware e software se dá pela Instruction Set Architecture (ISA) - conhecida como repertório de instruções em português, usando uma implementação em forma de linguagem de montagem (Assembly) gerada por compiladores de linguagens de programação de nível mais alto. Essa linguagem usa mnemônicos que abstraem a representação binária das instruções e de dados, de forma que seja mais fácil e intuitivo a programação da CPU. Os alunos da disciplina desenvolvem códigos em Assembly para entender como as instruções são executadas pelo processador, e consequentemente a forma que os compiladores geram essa linguagem a partir de códigos de nível superior. Ferramentas como simuladores de repertório de instruções são usadas para ler o código e gerar saídas com base nele, exibindo os dados nos registradores para verificação de correção. Geralmente essas ferramentas incluem um editor de texto próprio, além de outros elementos gráficos para guiar o usuário. Dado que existem poucas ferramentas similares, muitas delas não são multiplataforma e estão restritas a*

sistemas operacionais específicos, foi desenvolvido um interpretador de Assembly em formato de linha de comando de código aberto e propósito educacional. O objetivo é torná-lo multiplataforma e fácil de usar, sem a necessidade de usar um editor de texto específico. Isso facilitará o desenvolvimento das atividades da disciplina de Infraestrutura de Hardware e estará disponível gratuitamente para qualquer usuário interessado.

1. Introdução

Um computador é formado por duas partes que trabalham entre si para funcionar perfeitamente: *software* e *hardware*. Olhando mais detalhadamente para o *hardware* de um computador, temos vários componentes que desempenham seu papel para o funcionamento em conjunto, dentre eles a unidade de processamento ou *CPU*. A *CPU* é o componente principal que executa continuamente três ações: busca as instruções na memória, decodifica essas e por fim as executa [Patterson 2005]. Esse componente é responsável por executar as instruções presentes nos *softwares* que utilizamos no dia a dia para fazermos nossos deveres e também desfrutarmos de lazer.

Para que isto ocorra, é definido o repertório de instruções que cada CPU entende, cujo o termo em inglês é *Instruction Set Architecture* ou ISA. Um destes repertórios é visto em sala de aula na disciplina de Infraestrutura de Hardware do curso de Ciência da Computação do Centro de Informática da UFPE, a arquitetura MIPS. O repertório da arquitetura MIPS é baseado em registradores de 32 bits [Patterson 2005]. Por motivos acadêmicos o repertório tem como um de seus usos ensinar os alunos como este e outros repertórios funcionam fazendo a ponte entre hardware e software.

Atualmente a disciplina tem uma atividade para os alunos codificarem instruções deste repertório para ver de forma ilustrativa o funcionamento destas, aprendendo o fluxo de comandos e o que a unidade de processamento faz a partir do código de alto nível desenvolvido por programadores. A atividade é feita por um *software*, chamado MARS - *MIPS Assembler and Runtime Simulator* [Pete Sanderson 2006], que interpreta as instruções e simula o comportamento da unidade de processamento.

Contudo, o MARS possui certos pontos a melhorar, como por exemplo a atualização de sua interface gráfica que tende a ser mais compatível com o sistema operacional *Windows*, além do fato de possuir pouca documentação [Pete Sanderson 2006]. Ferramentas análogas são escassas e também apresentam uma dependência forte em plataformas e sistemas operacionais específicos, implicando de certa forma na obrigatoriedade do uso desta ferramenta. Além disso, a maioria das ferramentas não são mantidas e atualizadas de forma adequada, podendo sofrer com a falta de performance.

Dito isto, o simulador construído no formato de interface de linha de comando Grace propõe-se a ser uma ferramenta alternativa para o uso dos alunos da disciplina. Grace é uma alternatica para os alunos que são afetados pelo problema da ferramenta atual não ser multiplataforma ou que simplesmente queiram usar uma ferramenta diferente para executar as atividades da disciplina.

Adicionalmente a ferramenta proposta é uma solução pensada para ser mantida e atualizada com mais frequência e usando as tecnologias mais recentes para sofrer menos

com a perda de performance inerente aos softwares com o passar do tempo. Sendo estas ações fruto de uma das premissas que a ferramenta possui que é ter o código aberto e auditável para assim ter a capacidade de ser mantida e atualizada com mais frequência.

2. Contexto

A ISA MIPS é uma arquitetura de conjunto de instruções que define como um processador MIPS executa instruções de máquina. É uma arquitetura RISC (Reduced Instruction Set Computer) que tem o intuito de simplificar a execução de instruções, minimizando o número de ciclos de relógio necessários para executar cada instrução.

MIPS é caracterizada por uma ampla variedade de instruções que podem ser executadas em paralelo por um processador multiciclo em pipeline, além de um conjunto de registradores e formatos de instruções bem definidos. Além disso é frequentemente utilizada em sistemas embarcados, como roteadores, dispositivos móveis e sistemas de controle industrial, bem como em outros sistemas de alta performance.

Um simulador de MIPS é um software que permite que desenvolvedores ou estudantes de computação possam criar, depurar e testar programas escritos em linguagem assembly do MIPS sem a necessidade de um hardware real MIPS. Ele emula o comportamento do hardware MIPS e fornece uma interface gráfica do usuário para que o programador possa executar o código, visualizar os registradores do processador, o conteúdo da memória e as interrupções do sistema. Além disso, o simulador permite que o programador execute o código passo a passo, para que ele possa verificar a lógica do programa, monitorar as variáveis e depurar problemas, como loops infinitos ou instruções inválidas.

Existem diversos simuladores de MIPS disponíveis na web, alguns deles são gratuitos e de código aberto, enquanto outros são pagos e proprietários. Tomando como exemplo o MARS, o simulador usado atualmente na disciplina, sua estrutura consiste de um editor de texto integrado junto ao compilador de Assembly, onde existem elementos gráficos que ajudam o usuário a entender o funcionamento de um processador como por exemplo a lista de registradores e seus valores atuais [Pete Sanderson 2006].

Uma interface de linha de comando (CLI) é uma forma de interação com um computador ou sistema operacional por meio de texto digitado em uma linha de comando [Sampath et al. 2021]. Ao contrário das interfaces gráficas de usuário (GUI), que utilizam menus, ícones e janelas, a CLI é uma interface de texto simples que aceita comandos e responde com saídas de texto. A CLI é uma forma poderosa e flexível de interagir com sistemas operacionais e aplicativos de software, pois permite que os usuários realizem tarefas complexas por meio de uma série de comandos simples. Usuários avançados e programadores muitas vezes preferem a CLI porque ela pode ser mais eficiente e mais rápida do que a GUI para tarefas específicas.

Algumas das vantagens da CLI incluem comandos precisos e personalizados, seu maior controle e flexibilidade para tarefas complexas e a redução da dependência de interfaces gráficas complexas. Em resumo, a CLI é uma interface de texto simples que permite que os usuários interajam com computadores e sistemas operacionais por meio de comandos digitados em uma linha de comando. É uma ferramenta poderosa e flexível

para usuários avançados e programadores, permitindo que eles realizem tarefas complexas de forma eficiente e personalizada.

3. Motivação

O mote para a concepção deste trabalho, foi levada em consideração a aplicabilidade atual da ferramenta MARS na condução das atividades relacionadas ao entendimento da linguagem de montagem Assembly e como o processador interpreta essa linguagem para gerar a manipulação dos dados em memória.

A ferramenta atende as demandas básicas requeridas pela disciplina de Infraestrutura de Hardware, o sistema é bastante completo e poderoso com funcionalidades bastante pertinentes como por exemplo a lista interativa dos valores armazenados nos registradores [Pete Sanderson 2006].

Por ser um sistema desenvolvido por meio da linguagem de programação Java em sua versão 1.4.2 em meados dos anos 2000 [Pete Sanderson 2006], mais precisamente em 2006 por Kenneth Vollmar e Pete Sanderson da *Missouri State University*, e também pelo fato de ter sido fruto de uma pesquisa acadêmica, o sistema carece de manutenção para evitar obsolência inerente aos softwares feitos com tecnologias mais antigas. Um dos principais pontos que Grace pretende trazer como diferencial para o MARS é justamente a possibilidade de ser desenvolvidas novas funcionalidades por meio de contribuições características de um software livre.

A Figura 1 mostra a interface gráfica do MARS, com os elementos principais da ferramenta, como a tabela de valores dos registradores, junto a informações do programa Assembly em execução e a tabela de endereços de memória.

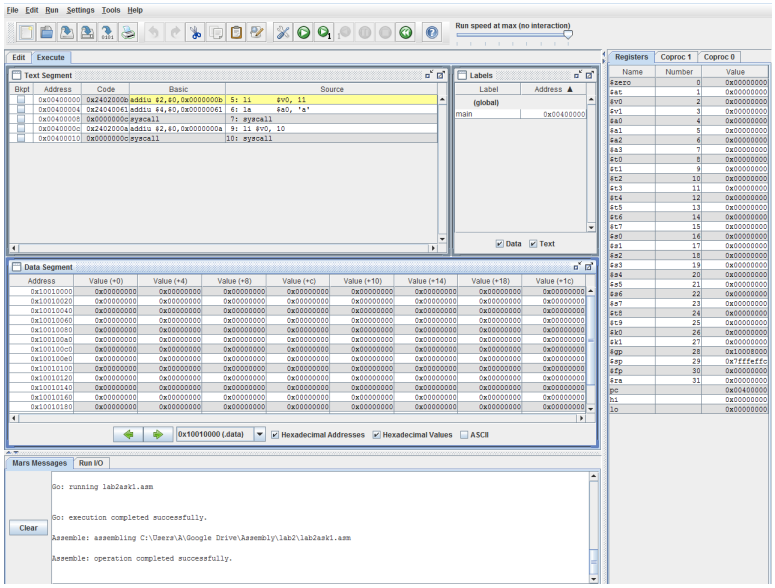

Figura 1. Interface do programa MARS

A interface gráfica do MARS é bastante completa, mostrando com simplicidade

vários elementos úteis para o usuário no momento da execução do programa. Contudo, a ferramenta poderia ter um fluxo de execução mais ágil em questão de interface, onde atualmente um usuário por exemplo ao querer visualizar os endereços de memória teria que clicar na tela várias vezes no botão com a seta verde abaixo da tabela.

Por outro lado, uma ferramenta que usa de uma interface de linha de comando, por mais que tenha que se ter um estudo mais elaborado de seus comandos, é potencialmente mais ágil na hora de apresentar os dados do programa para o usuário, sendo essa maior flexibilidade de execução de ferramenta um dos pontos a serem alcançados no desenvolvimento da solução proposta.

4. Grace

O sistema proposto para servir como alternativa à ferramenta MARS recebeu o nome de ”Grace”, em homenagem à cientista da computação Grace Hopper. Hopper é lembrada pelo pioneirismo em criar as primeiras linguagens de programação para computadores, que utilizavam comandos em inglês para tornar a programação mais acessível e compreensível [de Meira 2023].

A tecnologia empregada na construção desse sistema é o Java na sua versão 17, juntamente com o Maven como gerenciador de dependências e a biblioteca PicoCLI para estabelecer a interface de linha de comando. PicoCLI é uma biblioteca Java com a premissa de simplificar o desenvolvimento de interfaces de linha de comando, com elementos variados que são úteis para o programador, como por exemplo suporte a POSIX, GNU e MS-DOS, estas sendo normas para compatibilidade entre sistemas operacionais, e também conta com um esquema de colorização da saída dos comandos [Popma].

Com o objetivo de oferecer uma experiência de uso simplificada, o programa foi desenvolvido como uma Interface de Linha de Comando (CLI) que pode ser executada independentemente do sistema operacional, graças à portabilidade proporcionada pela Java Virtual Machine (JVM). O comando básico para uso da ferramenta na linha de comando é o *grace caminho-arquivo.asm*, onde passamos como argumento do comando o caminho do arquivo de instruções escolhido para ser simulado, com um exemplo de arquivo ilustrado na Figura 2. Atualmente, o número de linhas de código que o sistema possui é de cerca de 250, número que aumentará com a implementação de funcionalidades que ainda serão construídas.

Essa abordagem permite principalmente que estudantes de Infraestrutura de Hardware, ao se depararem com a necessidade de compreender e simular instruções de linguagem assembly MIPS, possam utilizar o simulador sem preocupações com incompatibilidades de plataforma. No cerne da sua funcionalidade, o simulador Grace oferece uma plataforma para a execução de conjuntos de instruções MIPS. Existem duas possibilidades de execução atualmente na ferramenta:

- primeiramente, temos a execução interativa, habilitada por padrão pela ferramenta, onde cada instrução é lida e executada uma a uma, com o resultado da

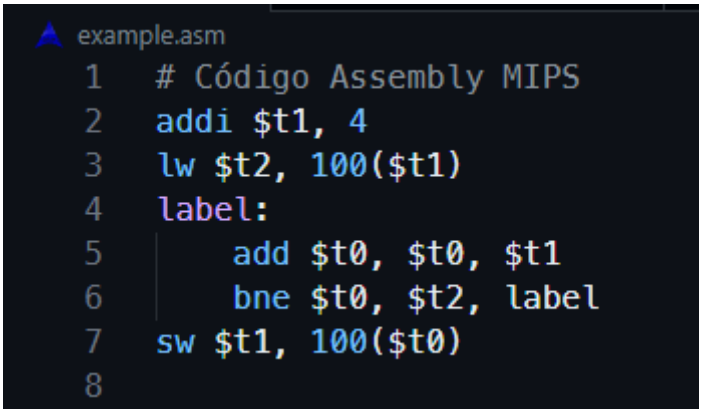

Figura 2. Exemplo de instruções MIPS

instrução em memória e nos registradores sendo escrito no terminal pelas funcionalidades da PicoCLI, seguindo o exemplo da Figura 3;

- alternativamente, ao usar a *flag* –run-all junto ao comando básico, a ferramenta irá ser executada de uma vez, gerando assim um arquivo de registro com cada passo executado pela leitura das instruções.

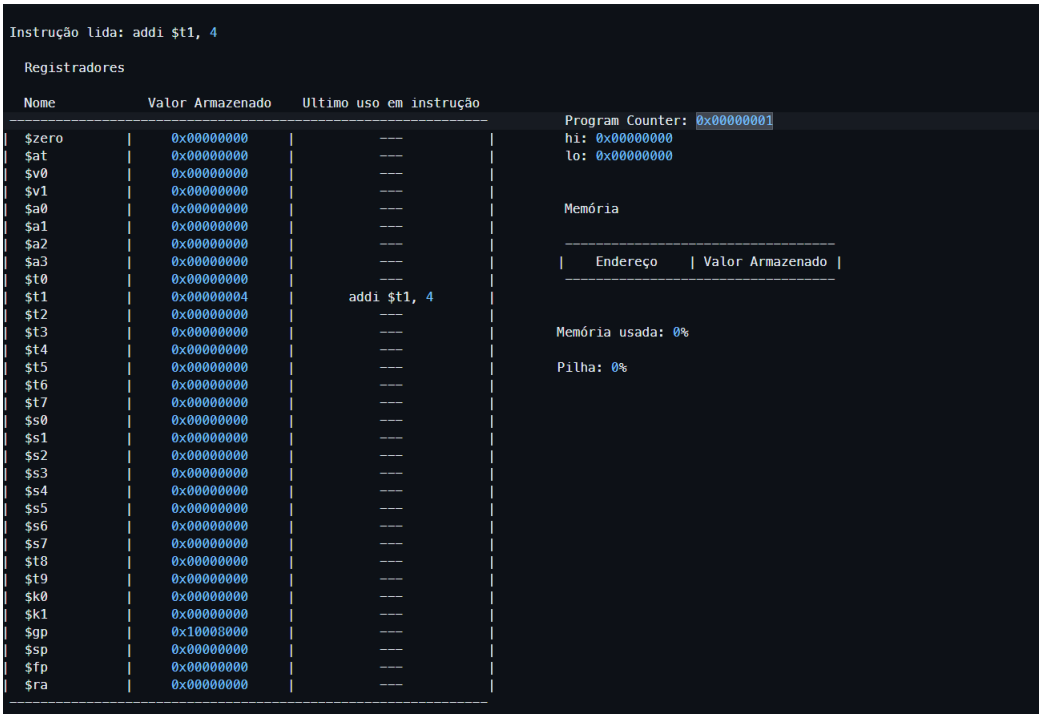

Figura 3. Exemplo de saída de dados da ferramenta Grace

Outras opções de comando que podem ser adicionadas ao comando básico são -D (ou –decimal) e -B (ou –binary), que se tratam de opções para mudar a base numérica que os dados armazenados nos registradores e na memória são apresentados para a base decimal ou para a base binária, respectivamente.

A ferramenta ainda está em fase de desenvolvimento, logo apenas um subconjunto de instruções presentes na arquitetura MIPS pode ser lida com sucesso. São as seguintes instruções aritméticas: *add, addi, sub, subi*; instruções de transferência de dados: *lw, sw*; e instruções de *jump: j, jal, beq e bne* [Gasparetto], fantando serem implementadas vinte e quatro instruções, além de treze chamadas de sistema que simulam o comportamento de dispositivos de entrada e saída para o usuário e sete diretivas usadas para armazenar dados na memória. Vale ressaltar ainda a possibilidade de ser desenvolvida por qualquer pessoa que sinta a vontade de contribuir para o projeto, abrindo assim um leque de possibilidades, como por exemplo dar suporte a repertórios de instruções mais complexos que sigam o padrão CISC [Patterson 2005].

A ferramenta faz o parsing, ou seja, a análise sintática de cada linha por vez, onde será lido o mnemônico da instrução para a sua correta execução, guardando a ordem da instrução no arquivo para o caso de haver instruções de jump ou de branching que possam alterar a ordem de execução do arquivo. Após essa fase, caso não ocorram erros de execução como por exemplo uma instrução inválida e caso o programa esteja executando na sua forma padrão, um snapshot das informações de registradores e da memória é escrito na tela do terminal, onde a execução fica parada até que o usuário aperte o botão Enter do teclado.

O tratamento de erros da aplicação é simples, ao capturar uma instrução inválida, o programa para a execução e apresenta em tela a mensagem de erro ”Não foi possível simular o arquivo”com uma mensagem personalizada com a instrução que ocasionou o erro e a linha correspondente a esta no arquivo.

Um ponto a se colocar sobre a ferramenta Grace é que pelo fato de ser uma CLI, possui independência de um editor de texto padrão, onde fica totalmente a cargo do usuário a escolha do editor de texto de sua preferência para escrever as representações das instruções Assembly a serem executadas pelo usuário.

Ao executar o conjunto de instruções desejado, o usuário também pode escolher o tipo de saída e relatório de corretude que pretende ter: o sistema foi projetado tanto para servir como um programa interativo ao longo da execução das instruções mostrando em linha de comando todos os dados disponíveis, seja de valores armazenados em registradores ou em memória, que foram atualizados pelas instruções, como pode também ser executado de uma vez e mostrado o passo a passo feito pelas instruções por meio de um arquivo de registro.

5. Estudo Comparativo

Após o desenvolvimento da aplicação, seguindo um subconjunto de funcionalidades em relação ao que foi proposto inicialmente, para fins de pesquisa acadêmica e validação da ferramenta Grace como alternativa ao MARS, foi usada a metodologia de estudo comparativo entre estas. Foram levados em consideração os seguintes critérios: flexibilidade, portabilidade e manutenção.

Antes de dar início ao estudo em si, será descrito cada critério e sua aplicabilidade na comparação entre o MARS e Grace. Em relação ao quesito flexibilidade e manutenção, o objetivo é comparar os detalhes técnicos de cada tecnologia por meio de testes de en-

trada e saída, a manutenção tem as particularidades de mensurar se o código pode ser mantido, estendido e melhorado continuamente.

O objeto dos testes será naturalmente um arquivo de instruções Assembly (declarado com a extensão *.asm*) com instruções que estejam contempladas no Grace para manter a paridade de funcionalidades. A portabilidade diz respeito ao quão disponível uma tecnologia é em quesito de plataformas. Ou seja, se a tecnologia funciona plenamente independente de sistemas operacionais.

5.1. Flexibilidade

Dando início ao estudo comparativo entre as duas ferramentas, talvez o principal ponto avaliador entre estas seja a flexibilidade. Ao analisar o MARS, é nítida a presença de elementos variados na interface gráfica. Pode-se apontar como exemplos a tela apresentando o conteúdo dos trinta e dois registradores que são atualizados de forma interativa a medida que as instruções são lidas, as palavras de 32 bits armazenadas em memória, além de um editor de texto embutido.

Já a solução Grace trabalha de uma forma diferente. A ferramenta Grace, por ser uma CLI, não possui uma interface gráfica tão apurada, porém a execução do sistema como um todo é mais performática por não necessitar de carregar os elementos em uma aplicação de interface gráfica no sistema.

Somado a esse ponto a ferramenta apresenta tanto a possibilidade de saída interativa no terminal com o usuário controlando a leitura e execução de cada instrução, com uma ilustração dos dados tão explicativa quanto a própria interface gráfica do MARS.

Existe também a possibilidade de geração de um arquivo de registros, este que apresenta tanto os valores em base hexadecimal, como também em base decimal e binária. Com disso, Grace nos apresenta alguns elementos do próprio MARS, este que serviu de inspiração para a construção do novo sistema, combinando-se também novos elementos visando assim um uso mais refinado e prático deste tipo de ferramenta.

Tomando como norte os pontos levantados acima, Grace e MARS indiscutivelmente cumprem o papel de executar código Assembly e apresentar ao usuário o conteúdo em memória após a leitura das instruções. Levando-se em conta a versatilidade de Grace em mostrar estes mesmos resultados, é possível afirmar que esta ferramenta leva vantagem em relação a ferramenta MARS no critério Flexibilidade.

5.2. Portabilidade

A portabilidade, como mencionado anteriormente, faz referência ao quão disponível em múltiplas plataformas um sistema está. Quando observamos a ferramenta MARS, ela funciona plenamente nos sistemas operacionais Linux, Windows e MacOs pelo fato do executável gerado funcione de maneira que qualquer sistema que tenha a JVM (Java Virtual Machine) instalada consiga usar a ferramenta.

Analogamente, a ferramenta Grace por ter sido desenvolvida em Java e executada como jar, possui a mesma característica da ferramenta MARS, sendo assim duas ferramentas que compartilham do mesmo nível de portabilidade.

5.3. Manutenção

No critério que diz respeito à manutenibilidade, os argumentos são mais favoráveis para Grace. Pelo fato da ferramenta MARS ter tido sua última atualização em 2016, o grau de manutenção da ferramenta é bastante aquém do necessário para qualquer tipo de software.

Mesmo sendo bastante funcional, o MARS carece de atualizações de interface e de algumas funcionalidades que poderiam ser otimizadas, como por exemplo a documentação explicativa do MARS que na verdade consiste em um resumo do artigo escrito por seus criadores. Além disto, ainda vale ressaltar o fato do código-fonte não estar acessível para outros desenvolvedores.

Já a ferramenta Grace tem como uma das premissas ser uma ferramenta open-source que possa ser mantida e estendida por vários programadores que tenham interesse para tal. O código será disponibilizado em um repositório Git [Conservancy] hospedado no gerenciador de código-fonte e controle de versão *Github*.

Tabela 1. Avaliação das ferramentas MARS e Grace.

Nome	Flexibilidade	Portabilidade	Manutenção
Grace	X	X	X
MARS		X	

6. Conclusão

Como Grace Hopper outrora foi referência na construção dos primeiros compiladores que serviram de rocha matriz para as linguagens de programação modernas, a ferramenta construída que leva seu nome propõe acima de tudo praticidade no decorrer do aprendizado dos alunos da disciplina de Infraestrutura de Hardware.

Possuindo como referência a Tabela 1, onde é mostrada a avaliação das ferramentas MARS e Grace, resultado do estudo comparativo entre as duas, podemos perceber que as ferramentas são bastante parecidas em diversos aspectos, com seus pontos fortes e fracos.

A Interface de Linha de Comando Grace cumpre o seu papel proposto na motivação, atingindo assim os objetivos estabelecidos de fornecer aos alunos de Infraestrutura de Hardware uma alternativa de software simulador de Assembly MIPS mais moderna a nível de implementação de código e com funcionalidades distintas do seu concorrente MARS.

6.1. Trabalhos futuros

Quando se vê o cenário atual da ferramenta, a primeira coisa a se pensar como trabalho futuro seria fazer as instruções presentes no repertório MIPS que ainda não foram implementadas, como também refinar a saída de dados de acordo com a utilização dos usuários finais do sistema.

Vale destacar também a possibilidade de disponibilizar a ferramenta num terminal integrado a uma aplicação web, ficando assim disponível para qualquer usuário com acesso a Internet.

Ao se ter um panorama futuro dos rumos que a ferramenta pode tomar, por sua destacada característica open-source, Grace pode ser cada vez mais melhorado e extendido a medida que for necessário por seus usuários, entusiastas do open-source e a quem mais interessar sobre essa área da computação que estabelece a sua base.

Referências

Conservancy, S. F. *Git. Disponível em https://git-scm.com/.* https://git-scm.com/.

de Meira, R. C. C. (2023). *Grace Hopper*.

Gasparetto. *Mips Instruction Set*.

Patterson, D. (2005). *Organização e projeto de computadores: a interface hardware/software*.

Pete Sanderson, K. V. (2006). *MARS: An Education-Oriented MIPS Assembly Language Simulator*.

Popma, R. *PicoCLI Disponível em https://picocli.info/.*

Sampath, H., Merrick, A., and Macvean, A. (2021). Accessibility of command line interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.