SIM-PROC: SOFTWARE EDUCACIONAL PARA O AUXÍLIO NO PROCESSO DE ENSINO-APRENDIZAGEM EM ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Fábio Weber Albiero[1]

Resumo
A compreensão de arquitetura e organização de microprocessadores é crucial para o desenvolvimento de sistemas computacionais modernos. No entanto, seu ensino enfrenta desafios devido à abstração desses conceitos e à falta de acesso a ferramentas adequadas. Este artigo introduz o SIM-PROC, um simulador de microprocessador projetado para melhorar o processo de ensino-aprendizagem nessa área. O SIM-PROC visa simplificar conceitos complexos, oferecendo aos estudantes uma plataforma virtual para visualizar o funcionamento interno de um microprocessador. Através de uma interface intuitiva, eles podem observar o estado dos principais registradores, da memória de dados e de instruções durante a execução de cada instrução. Além disso, os alunos podem personalizar parâmetros, conduzir experimentos e analisar resultados, tornando o aprendizado mais prático e eficaz.
Palavras-chave: Ensino-aprendizagem; microprocessador; simulador.

Abstract
Understanding microprocessor architecture and organization is crucial for the development of modern computing systems. However, teaching microprocessors faces challenges due to the abstraction of these concepts and the lack of access to appropriate tools. This article introduces SIM-PROC, a microprocessor simulator designed to improve the teaching-learning process in this field. SIM-PROC aims to simplify complex concepts by providing students with a virtual platform to visualize the inner workings of a microprocessor. Through an intuitive interface, they can observe the state of the main registers, data memory, and instructions during the execution of each instruction. Furthermore, students can customize parameters, conduct experiments, and analyze results, making learning more practical and effective.
Keywords: Teaching-learning; microprocessor; simulator.

1 INTRODUÇÃO

A disciplina de Organização e Arquitetura de Computadores representa um dos pilares do ensino da Computação, abordando princípios fundamentais sobre a estrutura interna e a comunicação dos componentes de hardware de um sistema computacional. Enquanto "organização" se refere aos componentes de hardware e suas interconexões, "arquitetura" diz respeito à construção do hardware, influenciando a execução lógica de aplicativos. No entanto, o ensino desses conteúdos pode ser desafiador devido à sua natureza técnica e abstrata.

Nesse contexto, a complexidade dos conceitos, como a estrutura dos componentes de hardware e a arquitetura dos microprocessadores, requer um sólido entendimento de conceitos matemáticos e físicos pelos alunos. Além disso, a percepção de que o conteúdo é tedioso e

[1] Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul-UFRGS; professor de Ensino Básico, Técnico e Tecnológico no Instituto Federal Farroupilha - campus Santo Ângelo. E-mail: fabio.albiero@iffarroupilha.edu.br.

monótono compromete o engajamento dos estudantes na disciplina. Tais dificuldades foram identificadas nos cursos de Licenciatura em Computação e Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia Farroupilha - campus Santo Ângelo/RS.

Para superar os desafios no ensino de Organização e Arquitetura de Computadores, os educadores devem adotar estratégias pedagógicas mais eficazes, incluindo atividades práticas e exemplos concretos para esclarecer conceitos abstratos. A integração de tecnologias educacionais oferece um recurso valioso ao permitir a identificação imediata dos erros cometidos pelos alunos, facilitando ajustes e melhorando a concentração, interpretação e raciocínio lógico por meio de feedback (Scattone; Masini, 2007). Essa combinação entre uma abordagem pedagógica adequada e o uso de recursos tecnológicos pode ajudar a mitigar os desafios no ensino dessa disciplina, tornando-a mais envolvente e relevante para os estudantes. Ademais, o eficaz ensino da arquitetura de computadores demanda a utilização de simuladores. Todavia, é pertinente destacar que muitos simuladores disponíveis no Brasil são complexos, carentes de documentação e apresentam interfaces gráficas pouco intuitivas, o que representa um desafio adicional para os educadores.

Diante deste contexto, o presente artigo apresenta o software educacional SIM-PROC (Programa de computador registrado no Instituto Nacional da Propriedade Industrial - Processo n° BR512018052159-2), projetado para simplificar o processo de aprendizado dos alunos em relação ao funcionamento dos microprocessadores e seus componentes. O SIM-PROC é uma ferramenta de simulação que oferece uma interface gráfica intuitiva, permitindo aos alunos visualizar o conteúdo dos principais registradores e da memória, assim como o fluxo de dados e de endereços de memória no decorrer da execução de cada instrução de máquina. Sua arquitetura é baseada nos processadores Intel 8080 (Intel Corporation, 1975) e 8086 (Intel Corporation, 1979; Intel Corporation, 1990); e a programação em Java permite a execução em diversos sistemas operacionais.

A solução proposta neste artigo representa uma ferramenta valiosa para educadores que buscam uma abordagem mais didática e interativa no ensino dos princípios de funcionamento dos microprocessadores e seus componentes, auxiliando no processo de ensino- aprendizagem.

O artigo está dividido em cinco seções: a seção 2 apresenta a fundamentação teórica, a seção 3 aborda os trabalhos correlatos, a seção 4 descreve o processo de desenvolvimento do aplicativo SIM-PROC, incluindo suas características e seu modo de funcionamento, e, por fim, a seção 5 apresenta as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta a fundamentação teórica deste trabalho, a qual consiste em uma revisão bibliográfica detalhada acerca dos principais conceitos e teorias relacionadas ao desenvolvimento do aplicativo SIM-PROC.

2.1 Microprocessadores e os seus componentes

O microprocessador, também conhecido como Unidade Central de Processamento (UCP), consiste em um circuito integrado responsável por realizar cálculos e tomar decisões em um computador. Suas principais funções incluem: 1) buscar as instruções na memória principal; 2) decodificar as instruções; 3) buscar e manipular os dados necessários; 4) executar as operações; 5) armazenar os resultados (se houver) e 6) reiniciar o processo para a próxima instrução (Monteiro, 2007; Patterson; Hennessy, 2013; Patterson; Hennessy, 2017).

O microprocessador é composto por uma série de componentes de hardware, cada um deles com funções específicas de processamento de dados ou de controle. De modo geral, para facilitar a organização dos componentes no interior do microprocessador, ele é dividido em Área Funcional de Processamento (AFP) e Área Funcional de Controle (AFC) (Monteiro, 2007). O barramento interno é representado pelo barramento de dados (BD), de endereços (BE) e de controle (BC), necessários para a conexão dos componentes internos da UCP. Os componentes presentes na AFC têm como função buscar e decodificar as instruções de máquina, enquanto os componentes da AFP se encarregam de executá-las.

A AFP da UCP é composta pela unidade lógica-aritmética (ULA) e pelos registradores. A ULA é o componente de hardware responsável por realizar as operações matemáticas básicas (adição, subtração, multiplicação e divisão) e operações lógicas, tais como: deslocamentos à direita e à esquerda, incremento e decremento de 1 etc. Já os registradores atuam como memória interna da UCP, armazenando os dados de modo temporário para que, posteriormente, sejam gravados na memória principal (Monteiro, 2007; Mueller, 2003; Stallings, 2017).

As funções de controle da UCP consistem em buscar a instrução de máquina e decodificá-la, além de gerar os sinais de controle apropriados para a ativação das atividades requeridas para a execução da instrução. A AFC é projetada para entender o que fazer, como fazer e comandar quem vai fazer o que, no momento adequado. A área funcional de controle é composta pelos seguintes componentes: o contador de instrução (CI), o decodificador de instrução, o registrador de dados da memória (RDM), o registrador de endereços da memória

(REM), o registrador de instrução (RI), a unidade de controle (UC) e o relógio (*clock*) (Monteiro, 2007; Patterson; Hennessy, 2013; Patterson; Hennessy, 2017).

2.2 A arquitetura do processador Intel 8080

O microprocessador Intel 8080 foi lançado em 1974. Ele possuía um conjunto de 72 instruções de máquina, 6 registradores de uso geral, tendo cada registrador 8 bits, barramento de dados bidirecional de 8 bits e um barramento de endereços unidirecional de 16 bits, o que permitia endereçar 64 kilobytes de memória RAM. O sinal do relógio trafegava por um barramento de 2 bits (Intel Corporation, 1975; Mueller, 2003).

De modo geral, as operações da ULA eram baseadas no uso de 3 registradores: o ACC, o FLAG e o TEMP, todos com 8 bits. Os bits do registrador FLAG eram definidos como sendo 0 ou 1, a fim de registrar algumas condições para o resultado da operação, por exemplo: caso o resultado fosse zero ou um número negativo (Intel Corporation, 1975).

O processador Intel 8080 possuía 6 registradores para uso geral, disponíveis para o programador, sendo eles: B, C, D, E, H e L. Tais registradores podiam ser utilizados isoladamente ou em pares (B/C, D/E e H/L), caso fosse preciso armazenar um dado com 16 bits, ficando os registradores B, D e H encarregados dos bytes menos significativos e C, E e L os bytes mais significativos. Para armazenar endereços de memória era utilizado o par de registradores H/L (ou registrador M) (Intel Corporation, 1975; Mueller, 2003).

2.3 A arquitetura do processor Intel 8086

Lançado pela Intel em 1978, o processador 8086 tinha um desempenho dez vezes melhor que o seu antecessor, o Intel 8080. O Intel 8086 continha 117 instruções básicas, seus registradores tinham largura de 16 bits, o BD tinha 16 bits de largura e o BE 20 bits, podendo endereçar mais de 1 milhão de bytes de memória (Intel Corporation, 1979; Intel Corporation, 1990). O microprocessador Intel 8086 era composto por duas unidades distintas, sendo a unidade de execução e a unidade de interface de barramento. A unidade de execução continha a ULA, os registradores de uso geral, enquanto a unidade de interface de barramento continha uma fila com as instruções, os registradores de comunicação interna e o apontador de instruções. A unidade de execução era composta por 8 registradores de uso geral (com 16 bits cada). Os registradores AX, BX, CX e DX eram divididos em dois segmentos, sendo o primeiro segmento (o mais à esquerda) o mais significativo, representado por AH, BH, CH e DH, e o segundo segmento o menos significativo, representado por AL, BL, CL e DL (Intel Corporation, 1979; Intel Corporation, 1990; Kant, 2014).

Ainda na unidade de execução havia o registrador FLAGS, o qual fazia uso de 6 bits para especificar determinadas condições para os resultados das operações. Os bits desse registrador eram: ZF (*Zero Flag*), SF (*Sign Flag*), TF (*Trap Flag*), IF (*Interrupt Flag*), DF (*Diretction Flag*) e OF (*Overflow Flag*) (Intel Corporation, 1979; Intel Corporation, 1990; Kant, 2014).

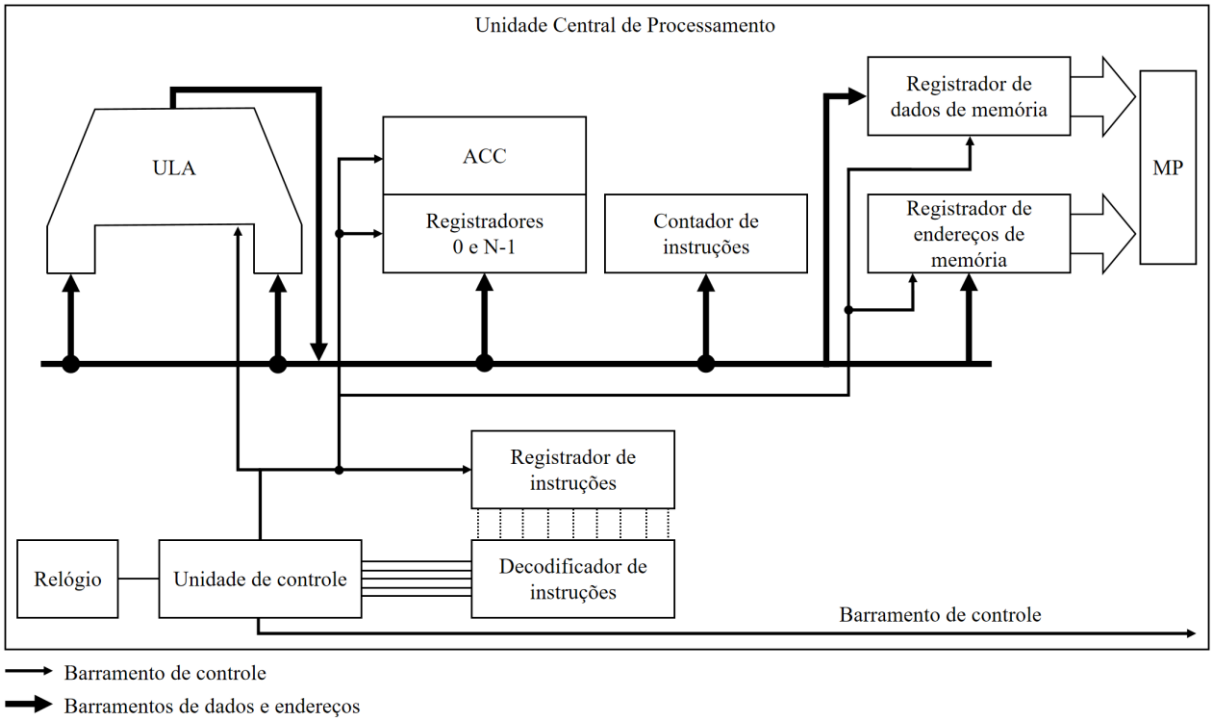
2.4 A arquitetura de um processador hipotético para fins didáticos

O microprocessador descrito por Monteiro (2007) foi concebido com o propósito exclusivo de ser utilizado como ferramenta didática. Sua arquitetura é delineada de forma simplificada na figura 1, onde se distinguem duas áreas funcionais principais: a AFP e a AFC. A AFP tem como principal atribuição a realização de atividades concernentes à efetiva execução das operações ou tarefas fundamentais do processador, como a execução de cálculos aritméticos ou operações lógicas. Ela é composta por unidades como a ULA, o registrador ACC e diversos registradores adicionais (R_0, R_1, …, R_{N-1}, onde N representa o número total de registradores desse tipo), destinados ao armazenamento de dados e instruções de máquina. A AFC, por outro lado, assume a responsabilidade de realizar operações como busca, decodificação e controle da execução das instruções de máquina, além de gerenciar as ações dos demais elementos do sistema. Os componentes que compõem a AFP e a AFC são conectados por meio de barramentos específicos (BD, BE e BC).

A maioria dos registradores possuem 8 bits, com exceção do registrador de instrução. Os registradores ACC e RDM podem armazenar dados e instruções, o RI armazena somente a instrução de máquina que será executada, enquanto CI e REM armazenam apenas endereços de memória. Diante do fato dos registradores CI e REM possuírem 8 bits, é possível endereçar 256 (2^8) células da memória principal, onde cada uma delas irá conter 12 bits (que é o tamanho de uma palavra e meia).

No que se refere ao formato das instruções, segundo o autor (Monteiro, 2007), elas são divididas em dois segmentos distintos: o primeiro, consiste no código da operação e define qual instrução será executada pela UCP, enquanto o segundo representa o operando, que pode variar de 1 a 3, dependendo da instrução. Além da quantidade de operandos que a instrução possui, pode variar também o modo de endereçamento dos dados, sendo eles: imediato, direto ou indireto. Monteiro (2007) utilizou na sua UCP instruções com apenas 1 operando e o modo de endereçamento direto. A quantidade de bits do código de operação é definida em função da quantidade de instruções que o processador possui. O autor apresenta um processador com 16 instruções. Portanto, são necessários 4 bits ($2^4 = 16$) para o código de operação, o que faz com que o RI tenha 12 bits (4 bits para o código de operação e 8 bits para o campo do operando).

Figura 1 – Esquema simplificado de uma arquitetura da UCP para fins didáticos.



Fonte: elaborada pelo autor (2025). Adaptação de Monteiro (2007).

Por fim, o ciclo de instrução apresentado na UCP é sequencial, ou seja, ela não faz uso da tecnologia de *pipelining*, processamento paralelo, fora de ordem, especulativo ou vetorial.

3 TRABALHOS CORRELATOS

No âmbito da simulação de microprocessadores, softwares educacionais desempenham um papel fundamental na compreensão e avaliação do funcionamento desses componentes cruciais nos sistemas computacionais. Eles replicam as interações complexas entre a arquitetura do microprocessador, a lógica de controle e a execução das instruções. Portanto, é essencial examinar os trabalhos relacionados para identificar lacunas no conhecimento e propor abordagens que aprimorem o processo de ensino em disciplinas como Organização e Arquitetura de Computadores. Nesta seção, será realizada uma revisão dos trabalhos correlatos, abordando as arquiteturas propostas para os microprocessadores e as perspectivas pedagógicas aplicadas no uso desses softwares.

Um dos trabalhos correlatos relevantes nesta área é o simulador NeanderWin (Borges; Silva, 2006), que oferece uma variedade de recursos, incluindo editor de textos, simulador da arquitetura, visualizador de memória simulada, simulador de visor, painel de chaves e várias ferramentas complementares para apoiar o aprendizado. Sua arquitetura é baseada na arquitetura Neander (Weber, 2000), com um conjunto de 16 instruções. Como trabalho futuro, os autores planejam introduzir uma instrução para movimentação de dados entre PC (*Program*

Counter) e ACC, implementar operações de deslocamento de bits, permitir a visualização online do fluxo de dados durante a execução das instruções e estender a arquitetura do processador para 16 bits. Do ponto de vista pedagógico, o NeanderWin apresenta limitações em usabilidade, como um esquema confuso de chaves e a exibição de valores em hexadecimal, o que requer que os alunos usem calculadoras para conversão de bases.

O MIPS-PMS (Elias; da Silva; Tiola, 2011) oferece uma simulação precisa do processador MIPS Multiciclo (Patterson; Hennessy, 2013), com uma interface intuitiva, limpa e dinâmica, focada na execução de códigos em linguagem Assembly para o processador MIPS. Seu layout inclui três painéis: um para os registradores, outro para os registradores temporários e um terceiro mostrando as posições de memória com valores de dados e instruções. A imagem da arquitetura MIPS Multiciclo é atualizada a cada passo da execução das instruções, exibindo os componentes de hardware utilizados em cada etapa. O software permite a inserção direta ou carregamento de código em Assembly e oferece um campo para regular a velocidade (em segundos) dos ciclos de relógio da simulação. Apesar de sua ampla gama de recursos, esse simulador apresenta um layout complexo devido à presença de múltiplas conexões (barramentos) e portas lógicas nos componentes, resultando em uma interface confusa para o usuário.

Por sua vez, o simulador descrito por Mustafa (2013) incorpora características de microprocessadores com arquitetura RISC (*Reduced Instruction Set Computer*). Essa arquitetura fragmenta as instruções em unidades menores e mais simples, todas com tamanho padronizado, cada uma projetada para execução em apenas um ciclo de relógio. O simulador apresenta um *pipeline* com 5 estágios e memória *cache*. Ele oferece funcionalidades como a criação e inserção de programas para execução, além da capacidade de remover programas existentes. Além disso, permite a criação, edição, movimentação e remoção de instruções nos programas desenvolvidos dentro do simulador.

O NeanderSIM (Ullmann *et. al.*, 2014) é um simulador educacional projetado para replicar as operações do computador Neander (Weber, 2000), demonstrando o fluxo de dados através de seus componentes e barramentos. Distingue-se de outros simuladores, como o NeanderWin (Borges; Silva, 2006), por várias razões: 1) utiliza recursos gráficos de animação para representar a execução das instruções nos componentes do computador; 2) é uma ferramenta portável, acessível via web, eliminando a necessidade de instalação e empregando tecnologias atuais de programação e 3) fornece *feedback* aos alunos, alertando sobre erros durante a inserção do código-fonte e informando sobre os passos da execução das instruções por meio de mensagens textuais. O NeanderSIM tem uma única memória, embora seja

representada como duas para fins didáticos: memória de dados e memória de instruções. Sua arquitetura é composta por três módulos: 1) módulo de interface gráfica, responsável pela interação com o usuário para criação e simulação de código; 2) módulo gráfico, que cria e anima elementos visuais representando os componentes do Neander e 3) módulo algorítmico, contendo as classes para execução das instruções suportadas pelo NeanderSIM.

O simulador descrito por Jesus, Moreno e Chella (2014) foi inspirado nos simuladores LC-3 e Neander (Weber, 2000). Para utilizá-lo, é necessário ter conhecimento da linguagem Assembly, pois é através dela que o usuário realiza a inserção de dados no programa. Implementado em Java, o simulador oferece uma interface gráfica simples e intuitiva. Ele suporta um conjunto de 16 instruções, cada uma com 16 bits, dos quais os 4 bits mais significativos representam o código de operação e os bits restantes são para os dados - 6 bits para cada operando. No que concerne à memória, ela contém um espaço de endereçamento de 16 bits (ou 2^{16} = 65.536 posições de memória).

O simulador SimuS (Silva; Borges, 2016) foi desenvolvido para suportar a arquitetura do microprocessador hipotético Sapiens, uma extensão da arquitetura e conjunto de instruções da UCP Neander-X. O SimuS oferece um ambiente integrado de desenvolvimento, permitindo que os alunos editem, compilem, depurem e executem programas escritos em linguagem de montagem específica para o Sapiens. Além disso, o SimuS possibilita a verificação do funcionamento final do programa, fornecendo acesso a todos os elementos que compõem o estado do microprocessador. A criação do SimuS foi motivada por desafios enfrentados no software Neander-X, como o espaço de memória limitado, dificuldades com as chamadas de rotinas, limitações no modo de endereçamento de memória e a falta de suporte a aritmética multi-byte.

4 APLICATIVO SIM-PROC

O SIM-PROC é um software educacional projetado para simular aspectos do funcionamento do microprocessador hipotético proposto por Monteiro (2007). Dentre esses aspectos, podemos citar: estados dos principais registradores, da memória de dados e de instruções, e a organização dos componentes de hardware. Seu principal objetivo é facilitar o processo de ensino-aprendizagem dos alunos acerca dos microprocessadores e da hierarquia de memória, fornecendo suporte para a disciplina de Organização e Arquitetura de Computadores. Desenvolvido utilizando o conjunto de pacotes gráficos JavaFX (Oracle Corporation, 2025), o SIM-PROC oferece uma interface gráfica simples e intuitiva que permite aos alunos visualizar claramente os principais componentes da UCP, incluindo dados, endereços de memória e

instruções. Além disso, o software oferece várias funcionalidades, como a seleção de instruções a serem executadas, controle do momento da próxima execução e modificação dos dados na memória principal. O SIM-PROC é projetado para desktops e laptops e não é compatível com dispositivos móveis. Ele está disponível em versões em Português (pt-BR) e Inglês (en-US) e não requer instalação, pois foi desenvolvido em linguagem de programação Java, possibilitando sua execução em qualquer computador com JVM (*Java Virtual Machine*) instalado.

As subseções a seguir descrevem, respectivamente: 4.1 - a arquitetura do microprocessador, 4.2 - o formato da instrução de máquina e o modo de endereçamento, 4.3 - o conjunto de instruções, 4.4 - a definição das instruções de máquina, 4.5 - funcionamento do microprocessador, 4.6 - a interface gráfica e, por fim, 4.7 - as funcionalidades do aplicativo.

4.1 A arquitetura do microprocessador

A arquitetura do microprocessador do SIM-PROC é a RISC e foi concebida com base na microarquitetura proposta por Monteiro (2007), a qual é mostrada na figura 1. A UCP é dividida em AFP e AFC. A AFP inclui a ULA e o registrador ACC. Já AFC é composta pelos seguintes componentes: CI, decodificador de instrução, RDM, REM, RI, registrador STATUS e unidade de controle (UC).

A seguir são descritos os principais componentes do SIM-PROC:

- CI: armazena o endereço da próxima instrução de máquina a ser executada pela ULA. Seu conteúdo é automaticamente modificado logo no início do ciclo de instrução. O CI possui tamanho de 12 bis, podendo endereçar 4.096 células de memória (2^{12} = 4.096). Ademais, cada célula de memória armazena 16 bits, o que resulta em uma memória com tamanho de 65.536 bits (4.096 células de memória × 16 bits = 65.536 bits) ou 8 kilobytes. O registrador CI terá o seu conteúdo modificado se: houver incremento automático no ciclo de instrução; o usuário reiniciar o sistema ou houver instruções de desvio (*jumps* ou *branches*);
- decodificador de instruções: identifica qual instrução de máquina será executada na ULA. O SIM-PROC possui 16 instruções de máquina e cada uma possui um código de identificação único. O decodificador de instruções recebe na sua entrada um conjunto de 4 bits para identificar uma das 16 instruções de máquina possíveis;
- RDM: armazena dados de memória. O RDM possui tamanho de 16 bits. Este registrador é utilizado pela UCP e pela memória principal para a transferência de dados de memória. No SIM-PROC, o processo de transferir o dado da memória principal para a UCP consiste em armazená-lo no RDM (RDM ← MP[endereço]). O processo inverso (de transferir o

dado da UCP para a memória principal) não faz uso do RDM, mas do registrador ACC (MP[endereço] ← ACC);

- REM: armazena endereços de memória. O REM possui tamanho de 12 bits. Este registrador é utilizado pela UCP e pela memória principal para a transferência de endereços de memória;
- RI: armazena a instrução a ser executada pelo microprocessador. O RI possui tamanho de 16 bits;
- registrador STATUS: registrador especial de controle. Diferente dos registradores de dados e de instruções, o registrador STATUS possui 4 bits. Cada bit possui um significado, indicando o estado de alguns elementos referentes a operação realizada. O bit menos significativo (bit mais à direita), chamado de "bit ZERO", quando em 1, indica que o resultado da operação é zero. O segundo bit ("bit SINAL"), da direita para esquerda, quando em 1, indica que o resultado da operação é um número negativo. O terceiro bit ("bit OVERFLOW"), da direita para a esquerda, quando em 1, indica que ocorreu *overflow* (transbordamento) no resultado da operação. E, por fim, o bit mais significativo (bit mais à esquerda), chamado de "bit IGUAL", quando em 1, indica que o valor gerado pela ULA e o valor contido no ACC são iguais. Os bits do registrador STATUS podem ter seus valores zerados (sofrer um *reset*) pelo usuário, a qualquer momento e
- UC: possui a lógica necessária para realizar a movimentação de dados e instruções de/para o microprocessador. A UC realiza as operações mediante sinais de controle que são emitidas em instantes de tempos determinados. De modo geral, todos os ciclos de instrução possuem duração fixa e igual, e são originadas pelo relógio. Com o objetivo de facilitar o processo de ensino-aprendizagem acerca do funcionamento da UCP, os sinais de controle gerados pelo relógio foram substituídos por sinais de controle gerados pelo usuário, quando este pressiona o botão "Executar próxima instrução".

É importante ressaltar que, no contexto do SIM-PROC, a memória principal armazena apenas os dados. Nesse sentido, as instruções são armazenadas diretamente no processador, mais especificamente na memória de instruções.

4.2 Formato da instrução de máquina e modo de endereçamento

A instrução em linguagem de máquina do processador SIM-PROC possui 16 bits. Os 4 bits mais significativos da instrução representam o código de operação (chamado de "op"), que define qual instrução será executada, enquanto os 12 bits restantes representam o campo

endereço (denominado “end”) e que contém um endereço da memória RAM cujo valor do operando está contido nele.

Quanto ao endereçamento, este consiste no ato de identificar em qual posição da memória RAM o operando da instrução está localizado. O modo de endereçamento usado no microprocessador do SIM-PROC é o endereçamento direto. No endereçamento direto, os bits contidos no campo endereço especificam a posição de memória onde se encontra o operando. É importante destacar que o endereçamento direto tem uso restrito, pois a instrução sempre acessa exatamente a mesma posição de memória. Em contrapartida, o endereçamento direto possui fácil implementação, é um dos mais rápidos modos de endereçamento de memória e um dos mais fáceis de se compreender o funcionamento (Monteiro, 2007; Patterson, 2017). Diante destas características, este foi o motivo da sua escolha para o desenvolvimento deste trabalho.

4.3 Conjunto de instruções

O microprocessador do SIM-PROC possui um conjunto de 16 instruções. O quadro 1 exibe o conjunto de instruções, descrevendo as funções que são desempenhadas por cada uma. Observe que a maioria das instruções necessitam de um endereço de memória RAM para a sua correta execução. No quadro 1, é possível observar que o endereço de memória principal acompanha a instrução de máquina e é representado pela palavra “end”.

Para que as instruções LD, STR, ADD, SUB, CMP, AND, OR, XOR, BRE e BRL sejam executadas, a ULA necessita ter dois operandos. Nas instruções LD, STR, BRE e BRL, um operando é um dado e o outro é um endereço de memória. Nesse caso, o dado localiza-se no registrador ACC, enquanto o endereço de memória localiza-se na instrução (no campo “endereço de memória”). Por exemplo, se o usuário deseja salvar um dado na memória principal (instrução STR), é necessário que ele possua o dado a ser salvo, que consiste no primeiro operando e está contido no registrador ACC, e saiba o local onde o dado será salvo, que é o segundo operando e é um endereço de memória, ou seja, o “end” da instrução de máquina “STR end”. Para as instruções ADD, SUB, CMP, AND, OR e XOR, os dois operandos são dados. Da mesma forma que ocorre nas instruções anteriores, um dado localiza-se no registrador ACC, enquanto o outro localiza-se na instrução, através do seu endereço de memória. Tomando como exemplo a instrução de máquina ADD, ela realiza a adição do valor contido no registrador ACC (primeiro operando) com o valor contido no endereço de memória especificado na instrução (segundo operando). O resultado gerado a partir da execução da instrução ADD na ULA é armazenado no próprio ACC, ou seja, o valor anterior é sobrescrito.

Quadro 1 – Conjunto de instruções do microprocessador do SIM-PROC.

Instrução	Código de operação	Descrição
NOP	0000	Não realiza nenhuma tarefa.
CLR	0001	Limpa o registrador ACC.
JMP end	0010	Efetua um desvio incondicional.
LD end	0011	Transfere o valor contido no endereço de memória especificado na instrução para o registrador ACC.
STR end	0100	Transfere o valor contido no registrador ACC para o endereço de memória especificado na instrução.
ADD end	0101	Efetua a operação aritmética de adição do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
SUB end	0110	Efetua a operação aritmética de subtração do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
CMP end	0111	Efetua a operação lógica de comparação do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
NOT	1000	Efetua o complemento do valor contido no registrador ACC.
AND end	1001	Efetua a operação lógica AND do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
OR end	1010	Efetua a operação lógica OR do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
XOR end	1011	Efetua a operação lógica XOR do valor contido no ACC com o valor contido no endereço de memória especificado na instrução.
RSH	1100	Efetua o deslocamento à direita de 1 bit do valor contido no registrador ACC.
LSH	1101	Efetua o deslocamento à esquerda de 1 bit do valor contido no registrador ACC.
BRE end	1110	Efetua um desvio condicional para o endereço especificado se o bit ZERO do registrador STATUS for 1.
BRL end	1111	Efetua um desvio condicional para o endereço especificado na instrução se o bit SINAL do registrador STATUS for 1.

Fonte: elaborada pelo autor (2025).

Destaca-se que as instruções NOP, CLR, NOT, RSH e LSH não necessitam de endereço de memória - não estão acompanhadas da palavra “end” de acordo com a tabela 1. Nesse caso, essas instruções utilizam apenas os 4 bits mais significativos da instrução de máquina, os quais representam o “código de operação”. Os bits que representam o campo do endereço de memória são ignorados no processo de decodificação dessas instruções, até porque estão preenchidos com bits aleatórios. Isso ocorre devido ao fato da ULA não necessitar de nenhum operando para a execução das instruções NOP e CLR, e de necessitar de apenas um operando, sendo que este já se encontra disponível na UCP, mais especificamente no registrador ACC, para a execução das instruções NOT, RSH e LSH.

As instruções de máquina JMP, BRE e BRL também merecem destaque. A instrução JMP possui apenas um operando que é o endereço de memória para onde a execução do programa será desviada. Nesse endereço de memória há outra instrução que será executada. Enquanto isso, as instruções BRE e BRL dependem dos valores contidos nos bits ZERO e SINAL do registrador STATUS, respectivamente. No caso da instrução BRE, se o bit ZERO do registrador STATUS for 1, haverá um desvio condicional para o endereço de memória especificado na

própria instrução. Por outro lado, se o bit ZERO for 0, o desvio não ocorrerá. Já para a instrução BRL, se o bit SINAL for 1, haverá um desvio para o endereço de memória determinado na instrução; caso contrário, se for 0, o desvio não ocorrerá. Para essas instruções de desvio, é importante ressaltar que, caso o endereço de desvio seja inválido, o programa do usuário é desviado para a última instrução válida, ou seja, para a última instrução da fila de instruções. Além disso, essas três instruções, juntamente com a instrução NOP, não fazem uso do valor contido no registrador temporário ACC.

As instruções aritméticas e lógicas ADD, SUB, NOT, AND, OR, XOR, RSH e LSH podem alterar os bits condicionais ZERO e SINAL do registrador STATUS de acordo com o resultado produzido. As instruções ADD e SUB ainda podem alterar o bit condicional OVERFLOW do mesmo registrador de acordo com o resultado gerado.

4.4 Definição das instruções de máquina

As instruções de máquina do SIM-PROC estão disponíveis no arquivo "input.txt". Para adicionar, remover ou alterar instruções, o usuário deve editar o arquivo e executar novamente o SIM-PROC. Cada linha do arquivo deve conter uma única instrução de máquina, haja vista que, caso haja duas ou mais instruções na mesma linha, o SIM-PROC não irá computar aquela instrução. As instruções podem ser escritas em letras maiúsculas ou minúsculas, tendo em vista que o *parser* do SIM-PROC é "*no case sensitive*". Para as instruções que exigem operandos, os quais consistem em endereços de memória, é necessário acrescentar o caractere `$` antes do endereço de memória. A figura 2 apresenta um exemplo de conjunto de instruções presentes no arquivo "input.txt", exemplificando o formato adotado para a manipulação de instruções no SIM-PROC.

Figura 2 – Exemplo de conjunto de instruções no arquivo input.txt.

```
LD $0
ADD $10
        NOP

LD $13
SUB $4
```

Fonte: elaborada pelo autor (2025).

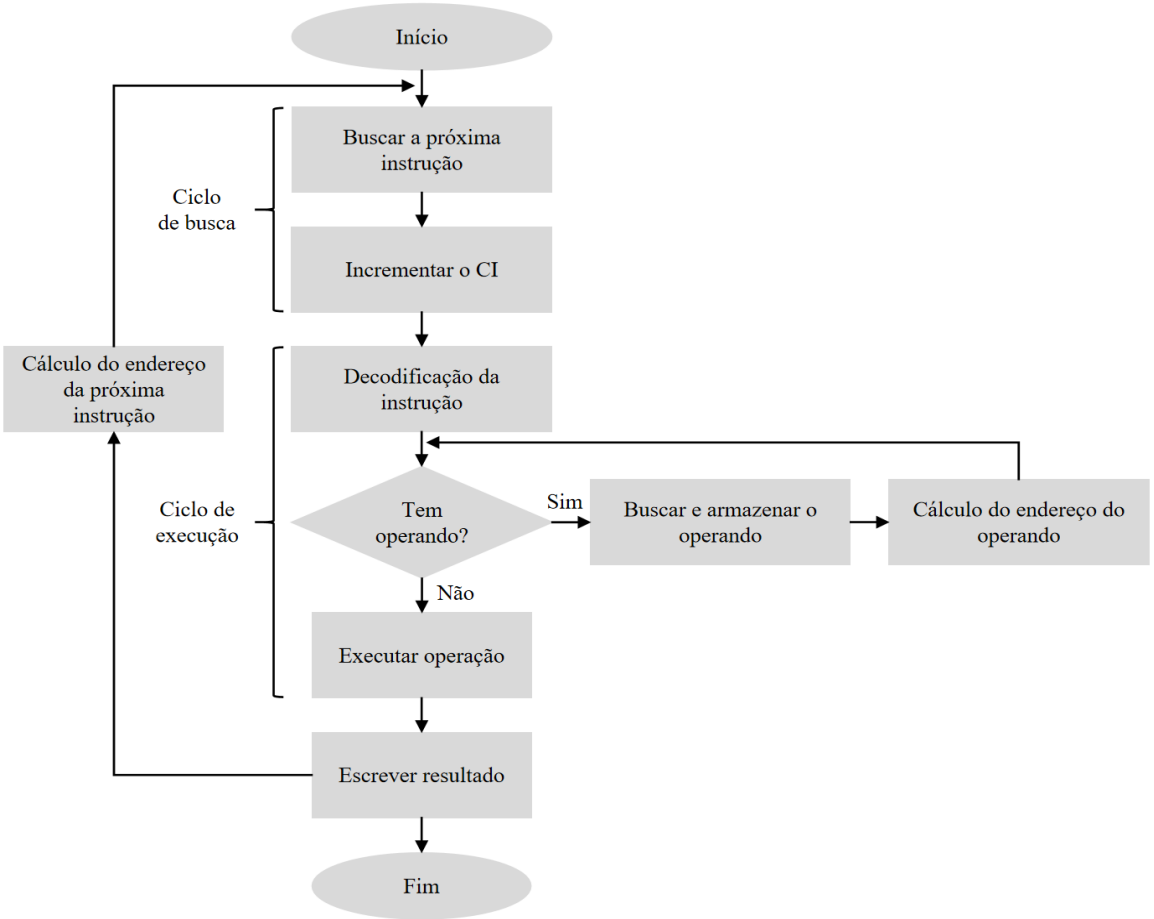
4.5 Funcionamento do microprocessador

O diagrama apresentado na figura 3 ilustra o fluxo de ações que compõe o ciclo de instrução do SIM-PROC. A primeira etapa a ser realizada consiste no ciclo de busca da instrução. Uma vez buscada a instrução, ela é transferida para o RI. Ao final da execução de

cada instrução, a próxima deve ser buscada, o que exige a atualização do CI, responsável por armazenar o endereço de memória da próxima instrução. Caso a instrução não seja de desvio (JMP, BRE ou BRL), o CI é simplesmente incrementado; caso contrário, assume o valor do operando especificado, que corresponde a um endereço de memória.

Após a busca, inicia-se a decodificação. O decodificador interpreta a instrução, separando os bits que compõem o campo "código de operação" dos bits que compõem o campo "operando". O campo "operando" consiste em um endereço de uma célula da memória principal que contém o dado. O dado é transferido para a UCP através de uma cópia para o RDM e representa o segundo operando da instrução a ser executada, enquanto o primeiro operando já se encontra no ACC. Se a instrução utilizar somente um operando, o processo de cópia do dado da MP para o RDM não é realizado. Ressalta-se que, na ausência de valor inicial no ACC (situação comum na execução da primeira instrução), e o usuário realizar alguma operação que não a operação LD (*load*), a UCP irá assumir que o valor contido no ACC é zero. Se o usuário executar a operação LD estará então carregando um dado da MP para o ACC, por meio do RDM.

Figura 3 – Diagrama com o fluxo de ações que compõe o ciclo de instrução do SIM-PROC.



Fonte: elaborada pelo autor (2025). Adaptação de Monteiro (2007).

A etapa final do ciclo de instrução corresponde à execução da operação e ao armazenamento do resultado no ACC, sobrescrevendo o valor anterior. Caso o usuário deseje salvar esse resultado na memória principal, deve utilizar a instrução STR (*store*), que transfere o conteúdo do ACC para o endereço de memória indicado no campo "operando" da instrução.

Para ilustrar esse funcionamento, considera-se a instrução "ADD $4". Segundo a notação adotada pelo simulador, o símbolo "$" indica um endereço de memória. Essa instrução realiza a adição do valor contido no endereço 4 da memória ao valor contido no ACC. Supondo que o ACC armazene o valor 10, o CI esteja em 2, a instrução "ADD $4" esteja localizada no endereço 2 da memória de instruções (figura 4a) e que o valor 20 esteja localizado no endereço 4 da memória de dados (figura 4b), o resultado da operação será 30, que será então armazenado no ACC.

Os passos que envolvem a execução da instrução ADD $4 são: 1) o valor contido no CI é copiado para o REM: CI = REM = 2; 2) o valor contido no CI é incrementado: CI = CI + 1; 3) a instrução localizada no endereço especificado por REM é buscada na memória de instruções e transferida para o RI: RI ← Mem[2]; 4) a instrução no RI é decodificada, revelando o "código de operação" ADD (0101, conforme a tabela 2) e o endereço de memória de dados (4); 5) esse endereço é copiado para o REM: REM = 4; 6) o dado localizado na memória de dados é copiado para o RDM e, posteriormente, para um dos registradores da ULA: RDM = 20 e R_0 = 20; 7) a ULA executa a operação: ACC = ACC + R_0, ou seja, ACC = 10 + 20 e 8) finalizada a instrução, o ciclo reinicia a partir do passo 1 com a próxima instrução.

Figura 4 – (a) Memória de instruções com a instruções ADD no endereço 2 e (b) memória de dados com o dado 20 no endereço 4.

0	
1	
2	ADD $4
3	
4	
…	…

(a)

0	
1	
2	
3	
4	20
…	…

(b)

Fonte: elaborada pelo autor (2025).

No SIM-PROC, a execução da instrução ocorre integralmente com um único clique no botão "Executar a próxima instrução", independentemente da quantidade de ciclos de instrução. Essa abordagem visa abstrair as rotinas internas da execução, simplificando tanto o processo de desenvolvimento do software quanto a visualização das operações pelo usuário. Além disso, contribui para o processo de ensino-aprendizagem ao tornar a ferramenta mais intuitiva e

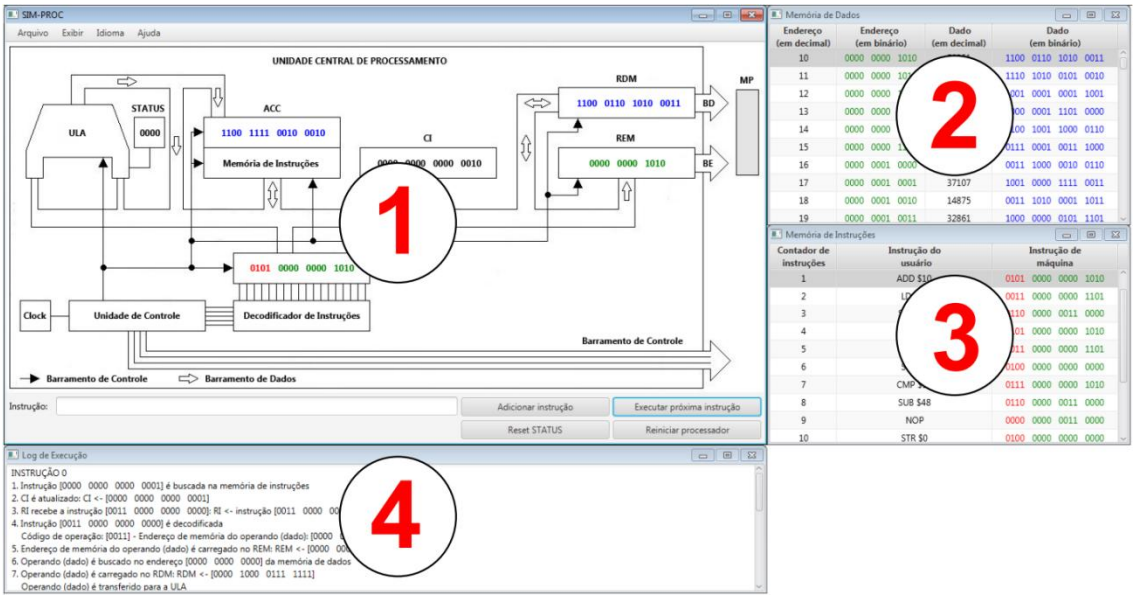
acessível, permitindo que os estudantes concentrem-se na lógica de funcionamento da UCP e na construção de programas.

4.6 Interface gráfica

O SIM-PROC apresenta quatro janelas: a janela principal, a memória de dados, a memória de instruções e o log de execução, como pode ser visto na figura 5. A janela principal, representada como janela 1 na figura 5, exibe os componentes internos do processador. Conforme as instruções são executadas, os valores armazenados nos registradores e em outros componentes são atualizados e apresentados ao usuário. Essa funcionalidade é fundamental, pois permitir que o aluno observe o comportamento do microprocessador durante a execução das instruções.

Por sua vez, as janelas 2 e 3 (figura 5) exibem as memórias de dados e de instruções. As informações armazenadas nessas memórias são representadas por meio de cores, a saber: azul para os dados, vermelho para o código de operação e verde para os endereços de memória. Isso permite a identificação clara e precisa das informações armazenadas nas memórias, facilitando a compreensão do funcionamento do sistema em questão. A figura 6 apresenta a janela da emória de dados (6. a) e a memória de instruções (6. b) com mais detalhes.

Figura 5 – Janelas do aplicativo SIM-PROC: 1 - janela principal; 2 - memória de dados; 3 - memória de instruções e 4 - log de execução.



Fonte: elaborada pelo autor (2025).

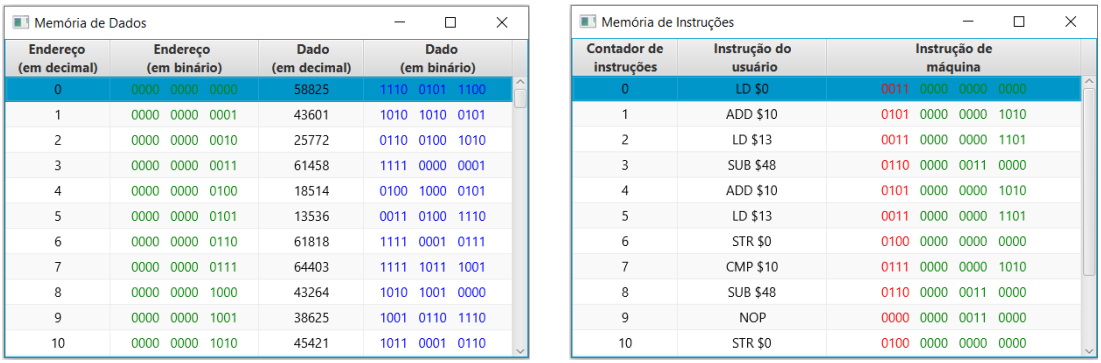
Por último, a janela 4 ilustra o log de execução (figura 5). Ela permite a identificação dos passos realizados durante a execução das instruções. A janela de log de execução é atualizada cada vez que uma instrução é executada, sendo que, quando o microprocessador é reiniciado,

toda a informação contida nessa janela é apagada. Cabe ressaltar que as informações presentes na janela de log de execução podem ser salvas em um arquivo de log (de extensão .txt) pelo usuário. Desta maneira, todas as informações relacionadas à execução das instruções são salvas no arquivo, possibilitando uma análise posterior mais detalhada das ações realizadas.

4.7 Funcionalidades do aplicativo

O SIM-PROC é uma ferramenta educacional que proporciona diversas funcionalidades para facilitar a compreensão do funcionamento de um processador e também da hierarquia de memória. As funcionalidades incluem a capacidade de selecionar quais instruções de máquina serão executadas na UCP, adicionar novas instruções durante a execução, executar uma instrução por vez e visualizar os valores armazenados em cada componente de hardware da UCP.

Figura 6 – (a) Janela da memória de dados e (b) janela da memória de instruções.



(a) (b)

Fonte: elaborada pelo autor (2025).

Ademais, o SIM-PROC permite editar os dados da memória principal (célula por célula) durante a execução, criar o log de execução, destacar os códigos de operação, os dados e os endereços de memória, zerar os bits do registrador STATUS (fazer um *reset*) e reiniciar microprocessador. A ferramenta oferece ainda a opção de exibir/ocultar a janela com os dados da memória principal, a janela com a memória de instruções e a janela com o log de execução. A possibilidade de alterar o idioma entre Português e Inglês também é uma funcionalidade do SIM-PROC, tornando-o uma ferramenta acessível e adaptável às diferentes necessidades.

É importante destacar também que o SIM-PROC pode ser utilizado em conjunto com outras técnicas e ferramentas para melhorar a compreensão no que tange o funcionamento dos microprocessadores e para auxiliar na solução de problemas em sistemas computacionais.

5 CONSIDERAÇÕES FINAIS

O artigo introduz o SIM-PROC, um simulador de microprocessador criado para aprimorar o processo de ensino-aprendizagem na disciplina de Organização e Arquitetura de Computadores. O trabalho detalha as características e funcionalidades do aplicativo, ressaltando sua relevância para a comunidade acadêmica.

O SIM-PROC é uma ferramenta educacional projetada para suprir uma carência no ensino de microprocessadores, proporcionando aos estudantes uma abordagem integrada entre a teoria e a prática. Com sua interface intuitiva e recursos avançados, o simulador oferece uma experiência imersiva, permitindo a compreensão do funcionamento interno de um microprocessador através da experimentação virtual. As vantagens notáveis incluem a habilidade de observar o estado dos registradores e das memórias durante a execução das instruções, bem como a flexibilidade para adaptar parâmetros e configurações. O SIM-PROC é um software gratuito, acessível a todos na comunidade acadêmica.

O SIM-PROC vem sendo utilizado na disciplina de Organização e Arquitetura de Computadores nos cursos de Licenciatura em Computação e de Tecnologia em Sistemas para Internet desde o ano de 2017.

Considerando o impacto positivo do SIM-PROC no processo de ensino-aprendizagem, acredita-se que ele possa ser amplamente adotado por outras instituições de ensino e acadêmicos interessados em promover uma educação mais prática e contextualizada. Ademais, o simulador SIM-PROC pode ser utilizado como ferramenta de pesquisa e desenvolvimento, permitindo a realização de experimentos e simulações em diversos cenários.

Como todo projeto, o SIM-PROC possui algumas limitações que podem ser aprimoradas em trabalhos futuros. Por exemplo, novas instruções de máquina poderiam ser adicionadas, aprimorando o conjunto de instruções já existente; assim como a tecnologia de *pipelining*.

Em resumo, o SIM-PROC representa um avanço significativo na área do ensino de microprocessadores, oferecendo uma abordagem prática para o processo de ensino-aprendizagem. Através da simulação e experimentação virtual, o simulador proporciona uma compreensão aprofundada de conceitos-chave, permitindo aos estudantes explorar, analisar e testar diferentes aspectos relacionados ao funcionamento dos microprocessadores.

REFERÊNCIAS

BORGES, J. A. dos S.; SILVA, G. P. da. NeanderWin: um simulador didático para uma arquitetura do tipo acumulador. In: WORKSHOP SOBRE EDUCAÇÃO EM ARQUITETURA DE COMPUTADORES (WEAC), 1., 2006, Ouro Preto. **Anais** [...]. [S. l.]:

Sociedade Brasileira de Computação, 2006. Disponível em: https://dcc.ufrj.br/~gabriel/WEAC2006.pdf. Acesso em: 28 jul. 2026.

ELIAS, W. J.; DA SILVA, J. R. C.; TIOLA, F. P. S. Simulador multiciclo do processador MIPS 32 bits para apoio ao estudo de arquitetura de computadores. **Revista Renote**, v. 9, n. 1, jul. 2011. Disponível em: https://doi.org/10.22456/1679-1916.21988. Acesso em: Acesso em: 28 jul. 2026.

INTEL CORPORATION. **Intel 8080 Microcomputer Systems User's Manual**. Set, 1975. Disponível em: http://www.bitsavers.org/components/intel/MCS80/98-153B_Intel_8080_Microcomputer_Systems_Users_Manual_197509.pdf. Acesso em: 28 jul. 2026.

INTEL CORPORATION. **8086 Family Users Manual.** Out, 1979. Disponível em: http://www.bitsavers.org/components/intel/8086/9800722-03_The_8086_Family_Users_Manual_Oct79.pdf. Acesso em: 28 jul. 2026.

INTEL CORPORATION. **8086 16-bit HMOS Microprocessor 8086/8086-2/8086-1.** Set, 1990, Disponível em: https://datasheets.chipdb.org/Intel/x86/808x/datashts/8086/231455-005.pdf. Acesso em: 28 jul. 2026.

JESUS, G. S.; MORENO, E. D.; CHELLA, M. T. **Simulador de Processador de Computador com Propósito Didático.** *In*.: XV Simpósio em Sistemas Computacionais de Alto Desempenho - WSCAD 2014, SBC, São José dos Campos - SP, p. 326-331, 2014.

KANT, K. **Microprocessors and Microcontrollers - Architecture, Programming and System Design 8085, 8086, 8051, 8096.** PHI Learning Private Limited, New Delhi, 2014.

SCATTONE, C. O; MASINI, E. F. S. O software educativo no processo de ensino-aprendizagem: um estudo de opinião de alunos de uma quarta série do ensino fundamental. **Revista Psicopedagogia**, [S. l.], v. 24, n. 75, p. 240–250, 2007. Disponível em: https://revistapsicopedagogia.com.br/revista/article/view/735. Acesso em: 28 jul. 2026.

MONTEIRO, M. A. **Introdução à Organização de Computadores.** 5. ed. LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro - RJ, 2007.

MUELLER, S. **Upgrading and Repairing PCs.** 14. ed., QUE. Indianapolis - IN, 2003.

MUSTAFA, B. YASS: A System Simulator for Operating System and Computer Architecture Teaching and Learning**.** *In*.: **European Journal of Science and Mathematics Education**, [S. l.], v. 1, n. 1, p. 34-42, 2013. Disponível em: https://www.doi.org/10.30935/scimath/9385. Acesso em: 09 jan. 2025.

PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores: A Interface Hardware/Software.** 5. ed. GEN LTC, 2017.

PATTERSON, D. A.; HENNESSY, J. L. **Arquitetura de Computadores - Uma Abordagem Qualitativa.** 5. ed. Elsevier Editora, Rio de Janeiro - RJ, 2013.

ORACLE COORPORATION. **JavaFX Overview.** Disponível em: https://www.tutorialspoint.com/javafx/javafx_overview.htm. 28 jul. 2026.

SILVA, P. G; BORGES, J. A. S. SimuS - Um Simulador para o Ensino de Arquitetura de Computadores**.** *In*.: **International Journal of Computater Archtecture Education (IJCAE)**, v. 5, n. 1, p. 7-12, 2016. Disponível em: https://www2.sbc.org.br/ceacpad/ijcae/v5_n1_dec_2016/IJCAE_v5_n1_dez_2016_paper_2_vf.pdf. Acesso em: 28 jul. 2026.

STALLINGS, W. **Arquitetura e Organização de Computadores.** 10. ed., Pearson Universidades, 2017.

ULLMANN, M. *et al*. E. NeanderSIM: simulador gráfico de apoio ao ensino de arquitetura de computadores. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 22., 2014, Brasília, DF. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2014. p. 1647-1656. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/10991. Acesso em: 28 jul. 2026

WEBER, R. F. **Fundamentos de Arquitetura de Computadores.** 1. ed., Editora Sagra Luzzato, 2000.