

UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

**João Vitor Pastori**

# RELATÓRIO TÉCNICO - Arquitetura e Org. de

**Computadores**

**APUCARANA – PR 2024**

**João Vitor Pastori**

# RELATÓRIO TÉCNICO - Arquitetura e Org. de Computadores

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

Professor: Guilherme Henrique de Souza Nakahata;

## APUCARANA–PR

**2024**

### SUMÁRIO

INTRODUÇÃO....................................................................................................... CAPÍTULO 1: OBJETIVOS....................................................................................

CAPÍTULO 2: MOTIVAÇÃO E RECURSOS UTILIZADOS....................................

2.1 Motivação.........................................................................................................

2.1 Recursos Utilizados.........................................................................................

2.2 Estrutura de Dados..........................................................................................

CAPÍTULO 3: RESULTADOS................................................................................

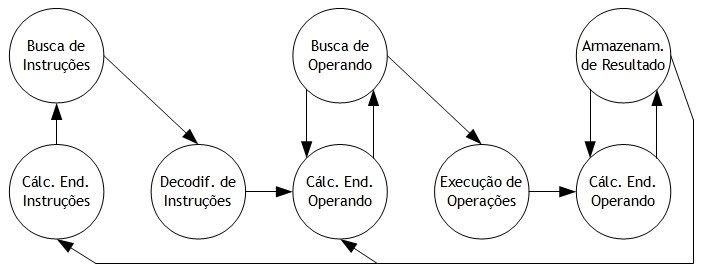
CONCLUSÃO........................................................................................................

### INTRODUÇÃO

Este relatório técnico descreve o desenvolvimento de um simulador de ciclos de instrução, criado como parte dos requisitos da disciplina de Arquitetura e Organização de Computadores do curso de Ciência da Computação. O objetivo do simulador é oferecer uma visão detalhada do ciclo de instrução, que abrange a busca, decodificação e execução das instruções, permitindo uma compreensão mais profunda do comportamento interno da Unidade Central de Processamento (CPU) durante a execução de programas.

A arquitetura de computadores é um campo crucial na ciência da computação, pois define a funcionalidade, organização e implementação dos sistemas de processamento. Compreender o ciclo de instrução é fundamental para construir uma base sólida nesse campo, possibilitando aos alunos visualizar como as instruções são traduzidas em operações de hardware. O simulador desenvolvido não apenas facilita essa compreensão, mas também serve como uma ferramenta prática para a experimentação e observação direta do ciclo de instrução.

Este trabalho visa não apenas atender aos requisitos acadêmicos da disciplina, mas também oferecer uma contribuição significativa para a compreensão prática da arquitetura de computadores. A Figura abaixo ilustra um ciclo de instruções da CPU:



### CAPÍTULO 1 OBJETIVOS

### CAPÍTULO 2 MOTIVAÇÃO E RECURSOS UTILIZADOS

O desenvolvimento do simulador de ciclos de instrução tem como objetivo oferecer uma ferramenta que permita aos alunos visualizar e entender detalhadamente o ciclo de instrução de uma CPU, abrangendo as etapas de busca de endereço, decodificação e execução das instruções. Um dos principais objetivos é facilitar a aplicação prática dos conceitos teóricos aprendidos na disciplina de Arquitetura e Organização de Computadores, possibilitando que os alunos experimentem e observem diretamente a interação entre os comandos e o hardware.

Além disso, o projeto estimula o desenvolvimento de habilidades de programação, promovendo a familiarização com linguagens e bibliotecas usadas no desenvolvimento de simuladores e ferramentas de análise de desempenho. O simulador também incentiva o pensamento crítico e a capacidade de criar soluções de engenharia para problemas complexos, por meio da construção e aprimoramento de uma ferramenta funcional. Finalmente, o projeto estabelece uma base para futuros estudos e desenvolvimentos na área de arquitetura de computadores, preparando os alunos para enfrentar desafios mais avançados e contribuir para a evolução da tecnologia de processamento.

Baseando-se no que foi anteriormente apresentado, é necessário apresentar os motivos e recursos utilizados para chegar ao resultado esperado.

**2.1 Motivação**

A criação deste simulador de ciclos de instrução é motivada pela necessidade de aprofundar a compreensão dos processos internos de uma Unidade Central de Processamento (CPU) durante a execução de programas. Ao oferecer uma visão detalhada do ciclo de instrução, que abrange as etapas de busca, decodificação e execução, pretende-se facilitar a compreensão dos conceitos teóricos de arquitetura de computadores. Esta ferramenta prática é fundamental para que os alunos visualizem como os comandos de software são convertidos em operações de hardware, consolidando uma base sólida em arquitetura de computadores, essencial para o progresso na área de ciência da computação.

Além disso, o simulador busca preencher a lacuna entre teoria e prática, proporcionando uma plataforma onde os estudantes possam experimentar diretamente os conceitos aprendidos em sala de aula. Ao analisar o comportamento do ciclo de instrução em diferentes cenários de teste, os alunos têm a oportunidade de identificar e compreender gargalos de desempenho, desenvolver soluções de otimização e fortalecer suas habilidades analíticas e de resolução de problemas.

**2.2 Recursos Utilizados:**

Para o desenvolvimento do simulador de ciclos de instrução, foram utilizados diversos recursos, tanto de software quanto de hardware, que são detalhados a seguir:

**2.2.1 Linguagem de Programação:** A escolha da linguagem de programação é fundamental para a implementação eficiente do simulador. Java foi selecionado devido à sua robustez, portabilidade e facilidade de manutenção. Além disso, Java oferece um vasto conjunto de bibliotecas que facilitam o desenvolvimento e a simulação de processos complexos, tornando-o uma escolha adequada para a construção de um simulador de ciclos de instrução.

**2.2.2 Ambiente de Desenvolvimento:** O simulador foi desenvolvido em um ambiente de desenvolvimento integrado (IDE) adequado, Visual Studio Code, que oferece ferramentas de depuração, gerenciamento de pacotes e suporte a extensões, facilitando o processo de codificação e teste.

Esses recursos combinados permitiram o desenvolvimento de um simulador funcional e eficiente.

**2.3 Estrutura de Dados**

O simulador de instruções implementado no código possui uma série de estruturas de dados e funções que permitem a execução e manipulação de um conjunto de instruções em um modelo simplificado de processador.

1. **Contador de Programa (contadorPrograma)**
   * **Tipo**: int
   * **Descrição**: Mantém a posição atual no programa, indicando qual instrução está sendo executada no momento.
2. **Registro Buffer de Memória (registroBufferMemoria)**
   * **Tipo**: int
   * **Descrição**: Armazena o valor temporário durante a execução das instruções, servindo como um buffer entre a memória e as operações aritméticas.
3. **Flags de Condição (flagZero e flagNegativo)**
   * **Tipo**: boolean
   * **Descrição**:
     + flagZero: Indica se o valor no registroBufferMemoria é zero.
     + flagNegativo: Indica se o valor no registroBufferMemoria é negativo.
4. **Memória (memoria)**
   * **Tipo**: int[] (array de inteiros)
   * **Descrição**: Armazena dados e valores intermediários com um tamanho fixo de 256 endereços (0 a 255).
5. **Lista de Instruções (instrucoes)**
   * **Tipo**: List<String>
   * **Descrição**: Armazena as instruções do programa a serem executadas, com cada instrução representada como uma string no formato "opcode op1 op2".

**Funções de Manipulação de Dados**

1. **instrucaoCarregarMemoria(int pos)**
   * Carrega o valor da memória na posição pos para o registroBufferMemoria.
2. **instrucaoSalvarMemoria(int pos, int dado)**
   * Salva o valor dado na posição pos da memória.
3. **instrucaoAdicionar(int pos)**
   * Adiciona o valor na posição pos da memória ao registroBufferMemoria.
4. **instrucaoSubtrair(int pos)**
   * Subtrai o valor na posição pos da memória do registroBufferMemoria.
5. **instrucaoMultiplicar(int pos)**
   * Multiplica o valor do registroBufferMemoria pelo valor na posição pos da memória.
6. **instrucaoDividir(int pos)**
   * Divide o valor do registroBufferMemoria pelo valor na posição pos da memória, com verificação para evitar divisão por zero.
7. **instrucaoPular(int pos)**
   * Altera o contadorPrograma para a posição pos, efetivamente pulando para a instrução na nova posição.
8. **instrucaoPularSeZero(int pos)**
   * Pula para a posição pos se o flagZero for verdadeiro.
9. **instrucaoPularSeNegativo(int pos)**
   * Pula para a posição pos se o flagNegativo for verdadeiro.
10. **instrucaoRaizQuadrada()**
    * Calcula a raiz quadrada do valor no registroBufferMemoria.
11. **instrucaoNegar()**
    * Inverte o sinal do valor no registroBufferMemoria.
12. **instrucaoSalvarBufferMemoria(int pos)**
    * Salva o valor do registroBufferMemoria na posição pos da memória.
13. **instrucaoNOP()**
    * Representa uma "nenhuma operação" (NOP), que não realiza nenhuma ação.

**Descrição das Instruções**

Cada instrução é representada como uma string no formato "opcode op1 op2", onde:

* **opcode**: Código da operação a ser realizada.
* **op1**: Primeiro operando, se aplicável.
* **op2**: Segundo operando, se aplicável.

Exemplos de códigos de operação e suas descrições:

* "000001": RBM <- #POS
* "000010": POS <- #DADO
* "000011": RBM <- RBM + #POS
* "000100": RBM <- RBM - #POS
* "000101": RBM <- RBM \* #POS
* "000110": RBM <- RBM / #POS
* "000111": JUMP to #LIN
* "001000": JUMP IF Z to #LIN
* "001001": JUMP IF N to #LIN
* "001010": RBM <- sqrt(RBM)
* "001011": RBM <- -RBM
* "001111": #POS <- RBM
* "001100": NOP

### CAPÍTULO 3 RESULTADOS

Os resultados da implementação do simulador de ciclos de instrução mostram que todos os objetivos foram alcançados com sucesso. O simulador consegue executar uma ampla gama de instruções, permitindo uma análise detalhada do ciclo de instrução de uma CPU. Durante os testes, verificou-se que os registradores, a memória e as flags funcionaram conforme o esperado, confirmando a precisão das operações realizadas pelo simulador.

Os testes abrangeram instruções para carregamento e armazenamento de dados, operações aritméticas e lógicas, além de controle de fluxo. Em todos os cenários, o simulador processou as instruções corretamente, atualizando os valores dos registradores e da memória de acordo com o especificado.

A interface interativa do simulador também facilitou a inserção e a visualização das instruções, oferecendo uma experiência prática e educativa para os usuários. A possibilidade de monitorar o estado dos registradores e da memória após cada instrução ajudou a aprofundar a compreensão dos conceitos teóricos da arquitetura de computadores.

Em resumo, o simulador desenvolvido atingiu os objetivos propostos ao fornecer uma ferramenta prática e eficaz para a visualização e execução de ciclos de instrução, contribuindo de forma significativa para o aprendizado e a compreensão da arquitetura de computadores.

### CONCLUSÃO

Com base na discussão, é claro que a implementação prática, mesmo em áreas mais teóricas, pode ser extremamente benéfica e adequada. Ela não apenas proporciona uma compreensão mais detalhada de conceitos externos à disciplina, mas também contribui significativamente para a fixação do conhecimento adquirido em sala de aula.

Durante o desenvolvimento e os testes, o simulador demonstrou ser robusto e eficaz, executando corretamente todas as instruções inseridas e atualizando os estados conforme as operações definidas. A interface interativa proporcionou uma experiência de aprendizado prática, permitindo que os usuários inserissem e visualizassem instruções de maneira intuitiva.

O projeto também ressaltou a importância de uma compreensão sólida dos conceitos teóricos de arquitetura de computadores, evidenciando como essas teorias são aplicadas na prática através de simulações. Observar diretamente o comportamento do ciclo de instrução ajudou a consolidar o conhecimento teórico e a desenvolver uma compreensão mais clara dos mecanismos internos de uma CPU.

Em resumo, o simulador de ciclos de instrução não apenas atendeu às expectativas iniciais, mas também se mostrou uma ferramenta valiosa para o ensino e aprendizado de arquitetura de computadores. Ele facilita a transição do conhecimento teórico para a aplicação prática, oferecendo uma plataforma onde os alunos podem experimentar e compreender profundamente o funcionamento de uma CPU.