



UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

**Theo Okagawa Rodrigues**

## **RELATÓRIO TÉCNICO - Arquitetura e Org. de Computadores**

**APUCARANA – PR  
2024**



**Theo Okagawa Rodrigues**

## **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

Neste relatório técnico, apresenta-se o desenvolvimento de um simulador de ciclos de instrução, elaborado como parte dos requisitos da disciplina de Arquitetura e Organização de Computadores do curso de Ciência da Computação. A criação deste simulador visa proporcionar uma visão detalhada do ciclo de instrução, que inclui a busca, decodificação e execução de instruções, permitindo uma compreensão aprofundada do comportamento interno de uma Unidade Central de Processamento (CPU) durante a execução de programas.

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

Este trabalho pretende não apenas cumprir os requisitos acadêmicos da disciplina, mas também fornecer uma contribuição significativa para o entendimento prático da arquitetura de computadores. A figura 1 abaixo representa um ciclo de instruções da CPU:

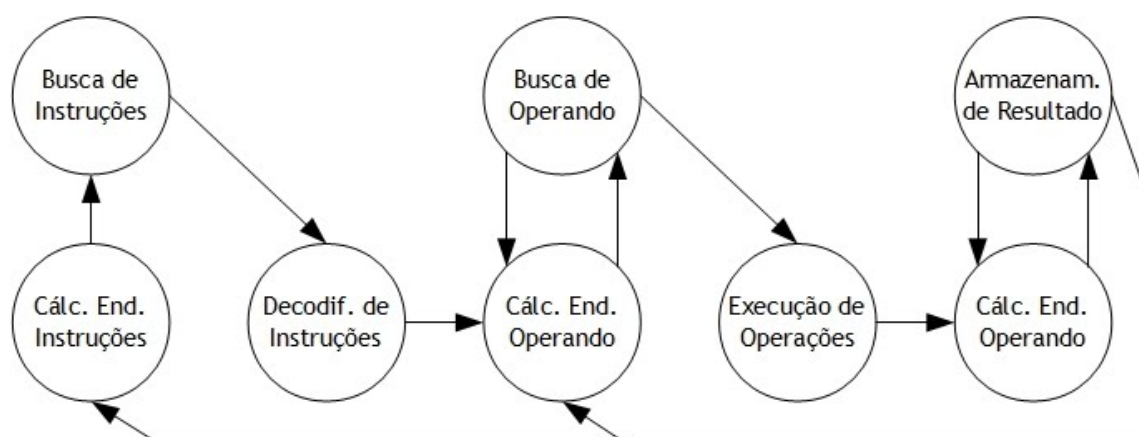


Figura 1

## CAPÍTULO 1

### OBJETIVOS

O desenvolvimento do simulador de ciclos de instrução visa proporcionar 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 objetivos centrais é facilitar a aplicação prática dos conceitos teóricos aprendidos na disciplina de Arquitetura e Organização de Computadores, permitindo que os alunos experimentem e observem diretamente como os comandos interagem com o hardware.

Além disso, o projeto incentiva o desenvolvimento de habilidades de programação, promovendo a familiarização com linguagens de programação e bibliotecas utilizadas no desenvolvimento de simuladores e ferramentas de análise de desempenho.

O simulador também estimula o pensamento crítico e a capacidade de desenvolver soluções de engenharia para problemas complexos, através da criação e aprimoramento de um simulador funcional. Por fim, o projeto serve como 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. A imagem abaixo (2) representa um exemplo de instruções, operandos e resultados de um ciclo de instrução:

Código da Instrução	Operandos	Resultado
000001	#pos	$MBR \leftarrow \#pos$
000010	#pos #dado	$\#pos \leftarrow \#dado$
000011	#pos	$MBR \leftarrow MBR + \#pos$
000100	#pos	$MBR \leftarrow MBR - \#pos$
000101	#pos	$MBR \leftarrow MBR * \#pos$
000110	#pos	$MBR \leftarrow MBR / \#pos$
000111	#lin	JUMP to #lin
001000	#lin	JUMP IF Z to #lin
001001	#lin	JUMP IF N to #lin
001010	-	$MBR \leftarrow \text{raiz\_quadrada}(MBR)$
001011	-	$MBR \leftarrow - MBR$
001111	#pos	$\#pos \leftarrow MBR$
001100	-	NOP

Figura 2

## **CAPÍTULO 2**

### **MOTIVAÇÃO E RECURSOS UTILIZADOS**

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

#### **2.1 Motivação**

A motivação para a criação deste simulador de ciclos de instrução reside na necessidade de aprofundar a compreensão dos processos internos de uma Unidade Central de Processamento (CPU) durante a execução de programas. Ao proporcionar uma visão detalhada do ciclo de instrução, que inclui as etapas de busca, decodificação e execução de instruções, espera-se facilitar a compreensão dos conceitos teóricos de arquitetura de computadores. Essa ferramenta prática é essencial para que os alunos visualizem como os comandos de software são traduzidos em operações de hardware, consolidando assim uma base sólida em arquitetura de computadores, um conhecimento crucial para o avanço na área de ciência da computação.

Além disso, o simulador tem como objetivo preencher a lacuna entre teoria e prática, oferecendo 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 podem identificar e entender os gargalos de desempenho, desenvolver soluções de otimização e, assim, 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 crucial para a implementação eficiente do simulador. Python foi selecionado devido à sua simplicidade, robustez e fácil compreensão e manutenção, além de possuir bibliotecas extensas que facilitam o desenvolvimento e a simulação de processos complexos.

**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.

**2.2.3 Documentação e Controle de Versão:** O controle de versão foi gerenciado através do Git, permitindo o rastreamento de alterações no código e a colaboração eficiente entre desenvolvedores. A documentação do código e do projeto foi mantida de forma organizada, utilizando ferramentas como Github para assegurar a clareza e a facilidade de entendimento.

Esses recursos combinados permitiram o desenvolvimento de um simulador funcional e eficiente, que não só atinge os objetivos acadêmicos propostos, mas também serve como uma ferramenta de aprendizado valiosa para os alunos do curso de Ciência da Computação.

## **2.3 Estrutura de Dados**

A estrutura de dados utilizada no desenvolvimento do simulador de ciclos de instrução foi projetada para representar os componentes principais de uma Unidade Central de Processamento (CPU) e a memória associada. Os principais elementos desta estrutura de dados são:

**2.3.1 Registradores:** São variáveis que armazenam informações temporárias

necessárias durante a execução das instruções.

- **PC** (Program Counter): Um contador de programa que mantém o endereço da próxima instrução a ser executada.
- **MBR** (Memory Buffer Register): Um registrador de buffer de memória que armazena dados que estão sendo transferidos para e da memória.
- **flagZero** e **flagNegativo**: Flags que indicam o estado dos resultados das operações aritméticas, sendo usadas para operações condicionais.


**2.3.2 Memória:** Representada por uma lista de tamanho fixo (**memoria = [0]\*256**), simula a memória RAM, onde os dados e instruções são armazenados. Cada posição da lista pode conter um valor de dado ou código de instrução.

**2.3.3 Instruções:** Armazenadas em uma lista (**instrucoes**), onde cada instrução é representada como uma string contendo o código de operação e seus operandos.

**2.3.4 Classe CicloInstrucao:** A classe **CicloInstrucao** encapsula a lógica do simulador, mantendo a estrutura de dados e métodos para manipulação e execução das instruções.

- **Inicialização (**\_\_init\_\_**)**: Configura os registradores, a memória e a lista de instruções.
- **Entrada de Usuário (**entrada\_usuario**)**: Coleta as instruções do usuário, adicionando-as à lista de instruções.
- **Visualização das Instruções (**ver\_instrucoes**)**: Exibe as instruções armazenadas, juntamente com seus códigos de operação e descrições.
- **Execução das Instruções (**executa\_instrucoes** e **rodar\_instrucoes**)**: Percorre a lista de instruções, decodificando e executando cada uma de acordo com o opcode.
- **Exibição do Ciclo (**exibe\_ciclo**)**: Mostra o estado atual dos registradores e a instrução sendo executada, para fins de acompanhamento do ciclo de instrução.



- 
- **Implementação das Instruções** (**inst000001**, **inst000010**, **etc.**): Métodos individuais para executar cada tipo de instrução com base no opcode.

Em resumo, a estrutura de dados do simulador é organizada para refletir de forma simplificada os componentes e operações de uma CPU real, permitindo uma simulação funcional do ciclo de instrução, facilitando o entendimento e a análise do comportamento da CPU durante a execução de programas.

## CAPÍTULO 3

### RESULTADOS

Os resultados obtidos com a implementação do simulador de ciclos de instrução demonstram que os objetivos estabelecidos foram plenamente alcançados. O simulador permite a execução de um conjunto variado de instruções, possibilitando a visualização detalhada do ciclo de instrução de uma CPU. Durante a execução, foi possível observar o comportamento correto dos registradores, da memória e das flags, confirmando que o simulador realiza as operações conforme esperado.

Os testes realizados incluíram instruções de carregamento e armazenamento de dados, operações aritméticas e lógicas, e controle de fluxo. Em todos os casos, o simulador executa as instruções corretamente, atualizando os valores dos registradores e da memória de acordo com as operações especificadas.

Além disso, a interface interativa do simulador facilitou a inserção e visualização das instruções, proporcionando uma experiência prática e educativa para os usuários. A capacidade de observar o estado dos registradores e da memória após cada instrução ajudou a consolidar a compreensão dos conceitos teóricos de arquitetura de computadores.

Em resumo, o simulador desenvolvido cumpre os objetivos de proporcionar uma ferramenta prática e funcional para a visualização e execução de ciclos de instrução, contribuindo significativamente para o aprendizado e a compreensão da arquitetura de computadores. A seguir, as imagens 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 e 13 demonstram um exemplo do pleno funcionamento do código:

```
> python -u "/home/okagawa/Documentos/unepar/Unepar_actv/2ano/c2ano/AOC/ciclo_instrucao/main.py"
=====
1 - Inserir
2 - Ver Instruções
3 - Executar
4 - Sair
```

Figura 3

```

Digite a opção: 1
Digite as instruções do programa (4 para sair da inserção)
Digite o código da instrução: 000010
Digite o primeiro operando: 251
Digite o segundo operando: 5
Digite o código da instrução: 000010
Digite o primeiro operando: 252
Digite o segundo operando: 10
Digite o código da instrução: 000010
Digite o primeiro operando: 253
Digite o segundo operando: 15
Digite o código da instrução: 000001
Digite o primeiro operando: 251
Digite o código da instrução: 000011

```

Figura 4

```

Digite o código da instrução: 000001
Digite o primeiro operando: 251
Digite o código da instrução: 000011
Digite o primeiro operando: 252
Digite o código da instrução: 000011
Digite o primeiro operando: 253
Digite o código da instrução: 001111
Digite o primeiro operando: 254
Digite o código da instrução: 4
=====

```

Figura 5


```

Digite a opção: 3
=====
EXECUTANDO
=====
=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000001

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000010
<OP1>: 251
<OP2>: 5

```

Figura 6



```
DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
251 <- 5
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000002

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000010
<OP1>: 252
<OP2>: 10
```

Figura 7

```
DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
252 <- 10
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000003

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000010
<OP1>: 253
<OP2>: 15
```

Figura 8

```

DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
253 <- 15
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000004

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000001
<OP1>: 251
<OP2>:

```

Figura 9

```

DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
MBR <- 251
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000005

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000011
<OP1>: 252
<OP2>:

```

Figura 10

```

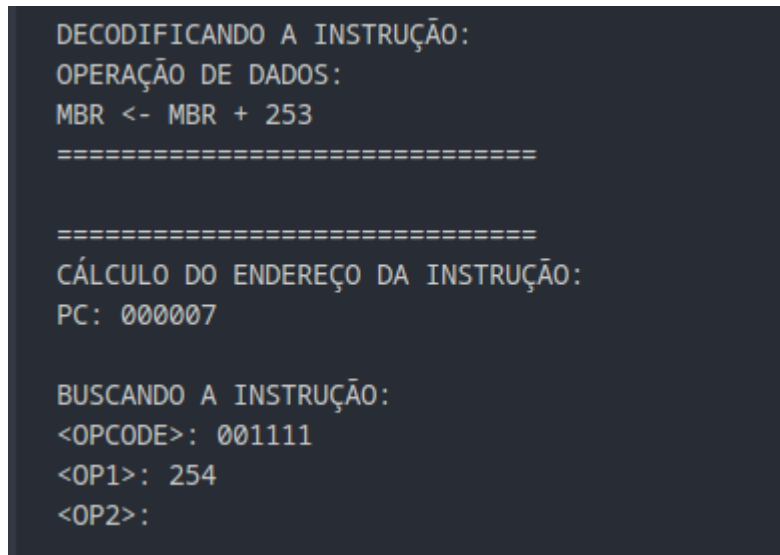
DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
MBR <- MBR + 252
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000006

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 000011
<OP1>: 253
<OP2>:

```

Figura 11

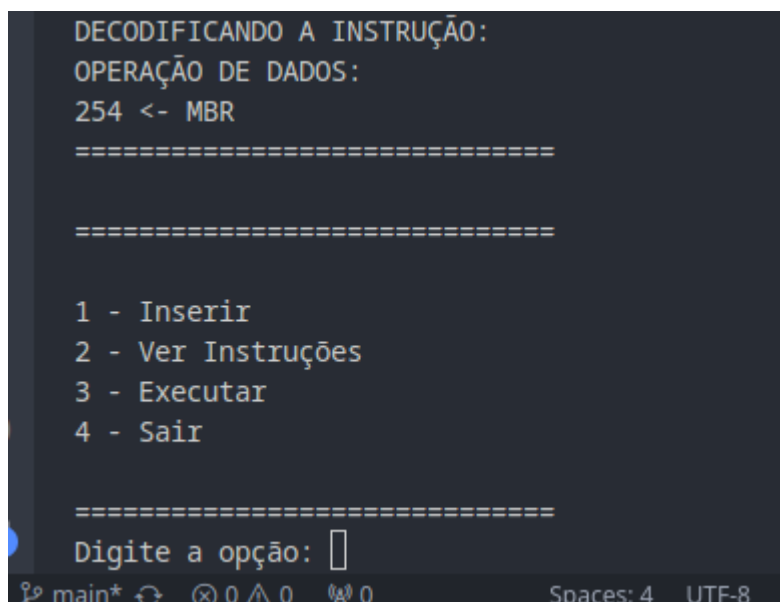


```
DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
MBR <- MBR + 253
=====

=====
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000007

BUSCANDO A INSTRUÇÃO:
<OPCODE>: 001111
<OP1>: 254
<OP2>:
```

Figura 12



```
DECODIFICANDO A INSTRUÇÃO:
OPERAÇÃO DE DADOS:
254 <- MBR
=====

=====

1 - Inserir
2 - Ver Instruções
3 - Executar
4 - Sair

=====
Digite a opção: 
```

main\* 0 0 0 Spaces: 4 UTF-8

Figura 13

## CONCLUSÃO

Com base no que foi discutido, é evidente que a implementação prática, mesmo em disciplinas mais teóricas, pode ser altamente vantajosa e apropriada. Ela não só proporciona uma compreensão mais profunda de conceitos externos à disciplina, mas também contribui significativamente para a consolidação do conhecimento adquirido em sala de aula.

Durante o desenvolvimento e testes, o simulador se mostrou robusto e funcional, executando corretamente todas as instruções inseridas e atualizando os estados de acordo com as operações especificadas. A interface interativa contribuiu para uma experiência de aprendizado prática, permitindo que os usuários inserissem e visualizassem instruções de maneira intuitiva.

O projeto também evidenciou a importância de uma compreensão sólida dos conceitos teóricos de arquitetura de computadores, destacando como essas teorias são aplicadas na prática através de simulações. A capacidade de observar diretamente o comportamento do ciclo de instrução ajudou a consolidar o conhecimento teórico e a desenvolver uma melhor compreensão dos mecanismos internos de uma CPU.

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