UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS APUCARANA*

Lissa Guirau Kawasaki

RELATÓRIO TÉCNICO - CICLO DE INSTRUÇÃO

Arquitetura e Organização de Computadores

APUCARANA – PR

2024

Lissa Guirau Kawasaki

RELATÓRIO TÉCNICO

Arquitetura e Organização 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

A disciplina Arquitetura e Organização de Computadores abrange diferentes áreas da computação e mecânica, estando presente em cursos técnicos, engenharias e bacharelados. Trata-se de uma disciplina fundamental para a formação de profissionais em tais áreas, isto é, garante que os conceitos estudados em algoritmos, programação e outras matérias teóricas, como variáveis, vetores e ponteiros tenham seu nível de abstração reduzido e suas instruções sejam traduzidas em ações físicas, como armazenamento e manipulação de dados e interação de componentes, e sejam mais bem compreendidos ao estudarmos a arquitetura e funcionamento de sua principal plataforma. Através das aulas ministradas, a compreensão da matéria permite a compreensão da linguagem da máquina, uma habilidade crucial para o desenvolvimento de softwares eficientes, especialmente em áreas relacionadas a computação de alto desempenho.

O ciclo de instrução é um componente essencial da arquitetura de computadores. Refere-se ao processo de execução de instruções armazenadas na memória como dados do processador central. O processador pega cada instrução e a executa em um único ciclo. Este ciclo começa com o processo de recuperação da instrução, decodificação e execução, e o ciclo termina quando a instrução completa a execução. O loop de instrução é um processo repetitivo que requer várias etapas para ser executado. Primeiro, o processador busca as instruções na memória. Em seguida, ele decodifica a instrução e executa de acordo com a instrução fornecida. O comando pode ser uma operação de leitura/gravação de dados ou uma operação aritmética lógica. Após executar a instrução, o processador realiza uma operação de armazenamento para que os resultados da instrução possam ser armazenados na memória. É importante entender que este ciclo é necessário para cada instrução dada pelo usuário e que o processador não perca nenhuma etapa do ciclo de instrução. Isso ocorre porque as instruções são armazenadas em sequência e, se uma etapa for perdida, o computador

não conseguirá concluir as instruções corretamente, resultando em saídas incorretas ou com erros.

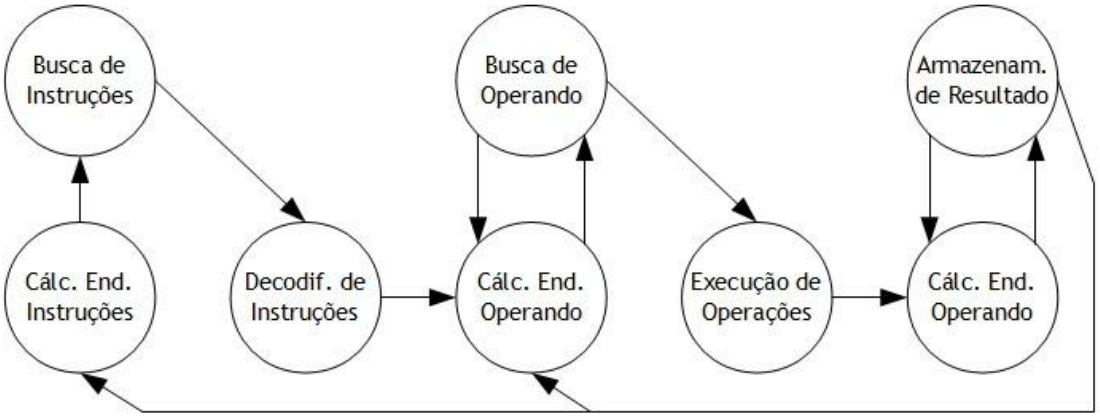


Figura 1, Conjunto de Instruções.

O ciclo de instrução é fundamental para a arquitetura moderna de computadores, afinal, todas as entradas do usuário exigem que a máquina complete o ciclo de instrução, ele também permite melhor desempenho da máquina, pois o processador pode executar instruções na mesma sequência para programas diferentes, resultando em melhores resultados. Dada a importância do loop de instruções, não é surpreendente que muitos computadores e linguagens de programação tenham sido projetados tendo o loop de instruções em mente. Isso permite que o processador execute comandos com mais rapidez, o que possibilita o desenvolvimento de programas de forma ágil e eficaz.

OBJETIVOS

O tema conecta conceitos teóricos a suas principais aplicações práticas, dessa forma, dentro da disciplina, cria-se a oportunidade de colocar em prática os aprendizados de linguagens de programação, realizando a simulação de um ciclo de instrução**.** O ciclo de instrução é uma parte fundamental do funcionamento dos processadores e sistemas de computação.

O objetivo de programar um ciclo de instrução é criar uma estrutura que permita ao processador executar tarefas complexas de maneira sequencial e organizada. Este ciclo é crucial para a execução de programas e algoritmos, que permitem ao computador realizar operações lógicas e aritméticas, manipular dados e controlar fluxos de execução. Em projetos de desenvolvimento e simulação de sistemas, compreender e programar um ciclo de instruções pode ajudar a otimizar o desempenho, resolver problemas e projetar sistemas de computador mais eficientes. Além disso, é uma habilidade importante para quem atua em arquitetura de computadores, compiladores e desenvolvimento de sistemas integrados. Em resumo, o loop de instruções é um mecanismo essencial que permite ao processador interpretar e executar comandos, realizar tarefas específicas e facilitar a execução de programas de forma ordenada e controlada.

MOTIVAÇÃO E RECURSOS UTILIZADOS

2.1 Motivação.

A principal motivação para análise e otimização de ciclos de instrução é a busca por maior desempenho em sistemas de computação, independentemente da plataforma. Em ambientes que vão desde computadores pessoais a servidores empresariais, a otimização do ciclo de instruções é essencial para maximizar a eficiência, velocidade e confiabilidade do programa.

Para profissionais e desenvolvedores, compreender e melhorar o ciclo de instrução é uma forma crítica de garantir o uso mais eficiente dos recursos computacionais. Com os avanços contínuos da tecnologia, comparar e avaliar o desempenho de sistemas e software tornou-se cada vez mais importante. Neste contexto, um ciclo de aprendizagem bem concebido serve como ferramenta de referência, permitindo uma comparação objetiva entre os diferentes sistemas. A flexibilidade do ciclo de instruções e sua capacidade de refletir o desempenho real do sistema tornam-no uma parte fundamental da avaliação de sistemas e software, a capacidade de otimizar o ciclo de instrução contribui para uma gestão mais eficiente e uma aplicação mais poderosa dos códigos.

Ao implementar e analisar o ciclo de instrução, podemos obter uma imagem clara das motivações e objetivos associados ao desempenho dos sistemas. Desta forma, uma análise detalhada da estrutura de dados e funções do ciclo de instrução permite uma melhor compreensão e visualização das estratégias de otimização e melhoria contínua.

2.2 Linguagem de Programação

Para o código, foi utilizado da linguagem de programação Python, uma linguagem que disponibiliza ferramentas eficientes e versáteis para o trabalho, a linguagem oferece uma gama de benefícios que impulsionam a produtividade e a resolução de problemas, sendo assim, a sua sintaxe clara e intuitiva e sua abundância de ferramentas torna ideal para a utilização em um programa que tem como seu principal objetivo a solução de problemas.

A biblioteca colorama é utilizada para adicionar cores ao output do terminal, tornando a saída mais visualmente atraente e fácil de entender.

2.3 Estrutura de Dados.

O código em Python simula o funcionamento básico de um processador, desde a carga de um programa até a execução de suas instruções. Ele demonstra os principais componentes de um processador e o ciclo de instrução, que é a sequência de passos que um processador realiza para executar uma única instrução.

I. Classe Processador:

A. Atributos:

1. Memória: Um array para simular a memória principal do computador.

2. Registradores: Um dicionário para armazenar os valores dos registradores (PC, IR, MAR, MBR).

Os registradores PC, IR, MAR e MBR são utilizados para controlar o fluxo de execução e armazenar dados temporários.

3. Flags: Um dicionário para armazenar as flags de status (Z e N).

As flags Z e N são utilizadas para indicar o resultado de operações aritméticas.

4. Instruções: Uma lista para armazenar as instruções do programa.

O simulador possui as seguintes instruções:

Código da Instrução	Operandos	Resultado
000001	#pos	MBR ← #pos
000010	#pos #dado	#pos ← #dado
000011	#pos	MBR ← MBR + #pos
000100	#pos	MBR ← MBR - #pos
000101	#pos	MBR ← MBR * #pos
000110	#pos	MBR ← MBR / #pos
000111	#lin	JUMP to #lin
001000	#lin	JUMP IF Z to #lin
001001	#lin	JUMP IF N to #lin
001010	-	MBR ← raiz_quadrada(MBR)
001011	-	MBR ← - MBR
001111	#pos	#pos ← MBR
001100	-	NOP

B. Métodos:

1. carregarPrograma: Carrega um programa (lista de instruções) na memória do processador.
2. buscarInstrucao: Busca a próxima instrução na memória e atualiza os registradores PC, MAR e IR.
3. Decodificar: Decodifica a instrução atual e retorna um objeto da classe de instrução correspondente.
4. executarInstrucao: Executa a instrução decodificada e atualiza as flags.
5. atualizarFlags: Atualiza as flags Z e N com base no valor do MBR.
6. exibirCiclo: Exibe detalhes do ciclo de instrução atual, incluindo o opcode, operandos e o resultado da execução.
7. exibirRegistradores: Exibe os valores atuais dos registradores e das flags.

II. Classe Instrução:
    A. Classe base para todas as instruções.
    B. Método executar: Método abstrato que deve ser implementado pelas subclasses para definir a ação de cada instrução.

III. Subclasses de Instrução:
    A. InstruçãoInserir: Insere um valor no MBR.
    B. InstruçãoVerInstrucao: Exibe os valores dos registradores.
    C. InstruçãoExecutar: Busca, decodifica e executa a próxima instrução.

IV. mapa_opcodes:
    A. Um dicionário que mapeia os códigos de operação (opcodes) para as classes de instrução correspondentes.

V. Classe ControlaCicloInstrucao:
    A. Atributos:
        1. processador: Uma instância da classe Processador.
        2. programa: Uma lista para armazenar as instruções inseridas pelo usuário.
    B. Métodos:
        1. entradaUsuario: Permite ao usuário inserir instruções e operandos.
        2. verInstrucoes: Exibe as instruções carregadas no programa.
        3. executarTodasInstrucoes: Executa todas as instruções do programa.

VI. Função usarCicloInstrucao:
    A. Função principal que cria uma instância da classe ControlaCicloInstrucao e oferece um menu interativo para o usuário.

Funcionamento:

1. O programa inicia e apresenta um menu para o usuário.
2. O usuário pode inserir instruções, visualizar as instruções carregadas e executar o programa.
3. A cada ciclo de instrução, o processador busca a próxima instrução, decodifica-a e executa a ação correspondente.
4. Os resultados da execução são exibidos no terminal, incluindo o estado dos registradores e as flags.

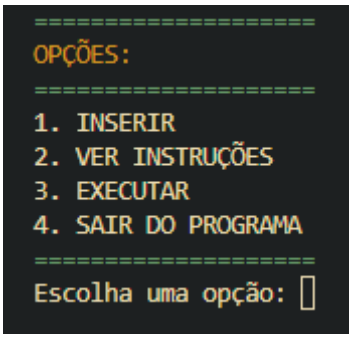
RESULTADO

Diante aos processos previamente realizados, o pleno funcionamento do código é atingido de forma esperada – para exemplificar o funcionamento do código, utilizaremos as seguintes operações:

Posição	Opcode	Operando1	Operando2	Operação
1	000010	251	5	Armazena 5 na posição 251
2	000010	252	10	Armazena 10 na posição 252
3	000010	253	15	Armazena 15 na posição 253
4	000001	251		MBR recebe o conteúdo da posição 251
5	000011	252		MBR recebe o conteúdo dele mesmo somado com o conteúdo da posição 252
6	000011	253		MBR recebe o conteúdo dele mesmo somado com o conteúdo da posição 253
7	001111	254		Posição 254 recebe o conteúdo de MBR
8	001100			Fim, no operation

As três primeiras instruções armazenam 5, 10 e 15, nas posições 251, 252 e 2 respectivamente. A quarta instrução carrega o registrador MBR com o conteúdo da posição 251 em seguida a próxima instrução realiza uma soma entre o conteúdo do registrador MBR com o conteúdo da posição 252 (10), atribuindo o resultado (15) ao registrador MBR. A instrução na posição 6 realiza uma nova soma entre o conteúdo do MBR (15) e o conteúdo da posição 253 (15) atribuindo o resultado (30) ao MBR. Por fim a instrução 7, salva o conteúdo do MBR (30) na posição 254 da memória, finalizando com a instrução 8 de nenhuma operação.

Inicialmente, no menu, escolhemos qual função realizar. Para a execução do exemplo citado acima, vamos prosseguir com a função “INSERIR”.



Em seguida, preenchemos as instruções do programa, assim como descrito na tabela.

```
Digite as instruções do programa (ou '4' para sair da inserção de dados):

Digite o código da instrução (em binário): 000010
Digite o primeiro operando: 251
Digite o segundo operando: 5

Digite o código da instrução (em binário): 000010
Digite o primeiro operando: 252
Digite o segundo operando: 10

Digite o código da instrução (em binário): 000010
Digite o primeiro operando: 253
Digite o segundo operando: 15

Digite o código da instrução (em binário): 000001
Digite o primeiro operando: 251

Digite o código da instrução (em binário): 000011
Digite o primeiro operando: 252

Digite o código da instrução (em binário): 000011
Digite o primeiro operando: 253

Digite o código da instrução (em binário): 001111
Digite o primeiro operando: 254

Digite o código da instrução (em binário): 001100
Digite o primeiro operando: 0
```

Inserindo o número “4”, podemos sair da inserção de dados e retornar ao menu principal, agora, podemos visualizar quais instruções foram preenchidas.

```
Instruções carregadas:
Instrução 1: Opcode 000010, Operandos [251, 5]
Instrução 2: Opcode 000010, Operandos [251, 5]
Instrução 3: Opcode 000010, Operandos [252, 10]
Instrução 4: Opcode 000010, Operandos [253, 15]
Instrução 5: Opcode 000001, Operandos [251]
Instrução 6: Opcode 000011, Operandos [252]
Instrução 7: Opcode 000011, Operandos [253]
Instrução 8: Opcode 001111, Operandos [254]
Instrução 9: Opcode 001100, Operandos [0]
```

Retornando ao menu principal, agora vamos executar as instruções, tendo como resultado o funcionamento das operações assim como descritas na tabela.

<ENDEREÇO DA INSTRUÇÃO ATUAL:>
PC: 000002

<CÓDIGO DA INSTRUÇÃO:>
IR <OPCODE>: 000010

<OPERANDOS:>
IR <OP1>: 251
IR <OP2>: 5

<DECODIFICANDO A INSTRUÇÃO:>
#POS <- #DADO
251 <- 5

<CÁLCULO DO ENDEREÇO DO OPERANDO:>
Endereço: 251

<BUSCANDO O OPERANDO NA POSIÇÃO:>
MAR: 251

<CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:>
Endereço: 5

<BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:>
MAR: 5

<OPERAÇÃO DE DADOS:>
    VALOR DO MBR: 0
    VALOR NA POSIÇÃO: 0
    VALOR DO MBR APÓS A OPERAÇÃO: 0 + 0 = 0
O VALOR FOI ARMAZENADO!

==========

PC: 2

IR: (2, [251, 5])

MAR: 1

MBR: 0

Flags: {'Z': True, 'N': False}

==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>

PC: 000003

<CÓDIGO DA INSTRUÇÃO:>

IR <OPCODE>: 000010

<OPERANDOS:>

IR <OP1>: 252

IR <OP2>: 10

<DECODIFICANDO A INSTRUÇÃO:>

#POS <- #DADO

252 <- 10

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 252

<BUSCANDO O OPERANDO NA POSIÇÃO:>

MAR: 252

<CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:>

Endereço: 10

<BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:>

MAR: 10

<OPERAÇÃO DE DADOS:>

VALOR DO MBR: 0

VALOR NA POSIÇÃO: 0

VALOR DO MBR APÓS A OPERAÇÃO: 0 + 0 = 0

O VALOR FOI ARMAZENADO!

==========

PC: 3

IR: (2, [252, 10])

MAR: 2

MBR: 0

Flags: {'Z': True, 'N': False}

==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>

PC: 000004

<CÓDIGO DA INSTRUÇÃO:>

IR <OPCODE>: 000010

<OPERANDOS:>

IR <OP1>: 253

IR <OP2>: 15

<DECODIFICANDO A INSTRUÇÃO:>

#POS <- #DADO

253 <- 15

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 253

<BUSCANDO O OPERANDO NA POSIÇÃO:>

MAR: 253

<CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:>

Endereço: 15

<BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:>
MAR: 15

<OPERAÇÃO DE DADOS:>
    VALOR DO MBR: 0
    VALOR NA POSIÇÃO: 0
    VALOR DO MBR APÓS A OPERAÇÃO: 0 + 0 = 0
O VALOR FOI ARMAZENADO!
==========

PC: 4
IR: (2, [253, 15])
MAR: 3
MBR: 0
Flags: {'Z': True, 'N': False}
==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>
PC: 000005

<CÓDIGO DA INSTRUÇÃO:>
IR <OPCODE>: 000001

<OPERANDOS:>
IR <OP1>: 251
IR <OP2>:

<DECODIFICANDO A INSTRUÇÃO:>
MBR <- #POS
0 <- 251

<CÁLCULO DO ENDEREÇO DO OPERANDO:>
Endereço: 251

<BUSCANDO O OPERANDO NA POSIÇÃO:>
MAR: 251

<OPERAÇÃO DE DADOS:>

VALOR DO MBR: 0

VALOR NA MEMÓRIA: 0

VALOR DO MBR APÓS A OPERAÇÃO: 0 + 0 = 0

O VALOR FOI ARMAZENADO!

==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>

PC: 000006

<CÓDIGO DA INSTRUÇÃO:>

IR <OPCODE>: 000011

<OPERANDOS:>

IR <OP1>: 252

IR <OP2>:

<DECODIFICANDO A INSTRUÇÃO:>

MBR <- MBR + #POS

251 <- 251 + 252

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 252

<BUSCANDO O OPERANDO NA POSIÇÃO:>

MAR: 252

<OPERAÇÃO DE DADOS:>

ARMAZENANDO:

NA POSIÇÃO: 252

<CALCULANDO ENDEREÇO DO OPERANDO:>

ENDEREÇO: 252

<ARMAZENANDO O OPERANDO:>

MAR: 252

<O VALOR FOI ARMAZENADO!>

==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>

PC: 000007

<CÓDIGO DA INSTRUÇÃO:>

IR <OPCODE>: 000011

<OPERANDOS:>

IR <OP1>: 253

IR <OP2>:

<DECODIFICANDO A INSTRUÇÃO:>

MBR <- MBR + #POS

251 <- 251 + 253

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 253

<BUSCANDO O OPERANDO NA POSIÇÃO:>

MAR: 253

<OPERAÇÃO DE DADOS:>

ARMAZENANDO:

NA POSIÇÃO: 253

<CALCULANDO ENDEREÇO DO OPERANDO:>

ENDEREÇO: 253

<ARMAZENANDO O OPERANDO:>

MAR: 253

<O VALOR FOI ARMAZENADO!>

==========

<ENDEREÇO DA INSTRUÇÃO ATUAL:>

PC: 000008

<CÓDIGO DA INSTRUÇÃO:>

IR <OPCODE>: 001111

<OPERANDOS:>

IR <OP1>: 254

IR <OP2>:

<DECODIFICANDO A INSTRUÇÃO:>

#POS <- MBR

254 <- 251

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 254

IR <OP1>: 254

IR <OP2>:

<DECODIFICANDO A INSTRUÇÃO:>

#POS <- MBR

254 <- 251

<CÁLCULO DO ENDEREÇO DO OPERANDO:>

Endereço: 254

<BUSCANDO O OPERANDO NA POSIÇÃO:>

MAR: 254

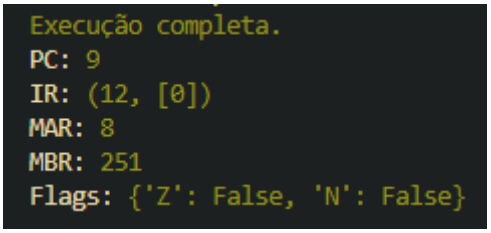
<OPERAÇÃO DE DADOS:>

==========

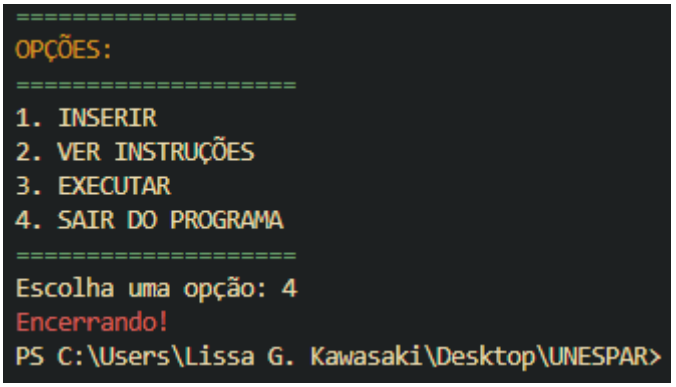
NOP

OPERAÇÃO FINALIZADA!

Ao finalizarmos a execução das instruções, temos o seguinte resultado:



Por fim, podemos retornar ao menu principal e finalizar o programa, selecionando a opção 4.



CONCLUSÃO

A análise e otimização do ciclo de aprendizagem desempenham um papel crucial na melhoria do desempenho dos sistemas computacionais. Ao compreender e aperfeiçoar cada etapa do ciclo de instrução, desde a busca e decodificação até a execução e armazenamento, é possível maximizar a eficiência, velocidade e confiabilidade dos programas, independente da plataforma utilizada. A implementação de um ciclo de aprendizagem eficaz não só contribui para o desempenho ideal do sistema, mas também fornece uma base sólida para comparações objetivas e benchmarking. O conhecimento desses tópicos é essencial para profissionais e desenvolvedores que precisam garantir que seus sistemas tenham o melhor desempenho em um mercado em constante mudança.

Além disso, a flexibilidade e a transparência na avaliação do ciclo educativo favorecem a inovação e incentivam a concorrência saudável no setor tecnológico. O domínio da otimização do ciclo de aprendizagem é essencial para solucionar problemas e identificar ineficiências no código, garantindo uma gestão mais eficiente e uma aplicação robusta de soluções.

Em resumo, o estudo e a aplicação do ciclo de instrução não só melhoram o desempenho dos sistemas, mas também fornecem informações valiosas sobre a estrutura e o funcionamento dos processadores, ajudando assim a alcançar a excelência na engenharia de software e contribuindo para o progresso contínuo da tecnologia.