**Aluno**: Felipe Cirne da Silva
**Matricula**: 2023016195

# Neander: Um Vislumbre da Arquitetura de Computadores

## Introdução

Criado para fins educacionais, o Neander proporciona uma visão clara de como as instruções são executadas, como a memória é manipulada e como as operações aritméticas e lógicas são realizadas em um nível de hardware básico.

## Arquitetura do Neander

A arquitetura do Neander é baseada em um conjunto reduzido de instruções (ISA - Instruction Set Architecture), o que simplifica o entendimento e a implementação de programas. A seguir, detalharei os componentes principais e o conjunto de instruções dessa arquitetura.

### Componentes Principais

1. Memória RAM: O Neander possui uma memória RAM que armazena tanto os dados quanto as instruções do programa. Essa memória é endereçável por byte, permitindo acesso direto a qualquer posição de memória.

2. Registradores: O Neander possui três registradores principais:
   - PC (Program Counter): Aponta para a próxima instrução a ser executada.
   - ACC (Acumulador): Utilizado para operações aritméticas e lógicas.
   - IR (Instruction Register): Armazena a instrução atualmente em execução.

3. ULA (Unidade Lógica e Aritmética): Responsável por realizar operações aritméticas (como soma e subtração) e lógicas (como AND e OR).

4. Barramento de Dados e Controle: Utilizado para transferência de dados entre os diferentes componentes do sistema.

**Aluno**: Felipe Cirne da Silva
**Matricula**: 2023016195

### Conjunto de Instruções

O conjunto de instruções do Neander é minimalista, composto pelas seguintes operações:

- LDA (Load Accumulator): Carrega um valor da memória no acumulador.
- STA (Store Accumulator): Armazena o valor do acumulador na memória.
- ADD (Add): Soma o valor da memória ao acumulador.
- SUB (Subtract): Subtrai o valor da memória do acumulador.
- JMP (Jump): Desvia a execução para um endereço específico.
- JN (Jump if Negative): Desvia a execução se o acumulador for negativo.
- JZ (Jump if Zero): Desvia a execução se o acumulador for zero.
- NOP (No Operation): Instrução que não realiza nenhuma operação.

## Exemplo de Implementação

A soma de dois números armazenados na memória em que que os números estão nas posições de memória 10 e 11, e o resultado deve ser armazenado na posição 12.

### Código Assembly para o Neander

```
LDA 10     ; Carrega o valor da posição 10 no acumulador
ADD 11     ; Soma o valor da posição 11 ao acumulador
STA 12     ; Armazena o resultado na posição 12
NOP        ; Instrução para finalizar o programa
```

### Interpretação do Código

1. LDA 10: O valor armazenado na posição de memória 10 é carregado no acumulador.
2. ADD 11: O valor armazenado na posição 11 é adicionado ao valor atual do acumulador.
3. STA 12: O resultado da soma, agora no acumulador, é armazenado na posição de memória 12.
4. NOP: Finaliza a execução do programa.

**Aluno**: Felipe Cirne da Silva
**Matricula**: 2023016195

## Conclusão

O Neander, com sua arquitetura simples e eficiente, oferece uma excelente base para o estudo e entendimento dos conceitos fundamentais de arquitetura de computadores. Através de seu conjunto reduzido de instruções e componentes claramente definidos, ele permite explorar o funcionamento interno de um processador, facilitando a compreensão de tópicos mais complexos em computação.

## Referências

- Tanenbaum, A. S. (2015). Organização Estruturada de Computadores. Pearson.
- Stallings, W. (2013). Arquitetura e Organização de Computadores. Pearson.
- Patterson, D. A., & Hennessy, J. L. (2014). Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann.