UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMAS DIGITAIS

**Processador embasado em máquina de Von-Neumann**

NATAL – RN

2017

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMAS DIGITAIS

**Processador embasado em máquina de Von-Neumann**

*Relatório do Processador de Von Neumann. Projeto criado para o curso de Sistemas Digitais na Engenharia Mecatrônica.*

**Professor:** Samaherni Morais Dias

**Aluno:** Iago Lucas Batista Galvão

NATAL – RN

2017

**Sumário**

**Introdução4**

**Objetivo5**

**Fundamentação Teórica6**

**Processador8**

**Resultados5**

**Conclusão5**

**Referências**

**1. Introdução**

Os primeiros computadores possuíam programas fixos. Assim, para alterar o programa de tal máquina era necessário reestruturar, re-projetar ou re-ligar os componentes. Esse processo era muito trabalhoso, pois exigira fluxogramas e cédulas de papel, desenhos técnicos detalhados, para após modificar o hardware da máquina. Todo o processo poderia levar semanas para começar a utilizar o programa desejado.

Computadores simples, ainda utilizam esse tipo de arquitetura. Pois, para fins de formação ou simplicidade, não é exigido a modificação dos programas. Como, por exemplo, calculadoras simples que possuem um programa fixo que realiza operações matemáticas, entretanto não processa arquivos de texto ou gráficos.

Com isso, Jonh Von-Neumann, junto a outros pesquisadores descobriram que, ao utilizar dispositivos de memória em forma de linha de retardo de mercúrio, são capazes de armazenar as instruções dos programas. O conceito de programas armazenado, seria a solução para o trabalhoso processo de modificação, pois era possível armazenar na memória, conjuntos de instruções que detalha o cálculo.

Ao amadurecer essa ideia, o projeto EDVAC foi criado e, a partir deste, o conceito de programa armazenado foi implementado, como também, era possível modificar programas durante sua execução.

Entretanto, havia uma desvantagem, o gargalo de von Neumann. Ao separar a memória da CPU, a taxa de transferência é afetada. Pois a CPU é colocada em espera para a ler ou escrever os dados na memória. Com o crescimento da memória e a velocidade da CPU vem aumentando mais do que a taxa de transferência entre eles, tal problema é aumentado junto ao advindo das gerações de CPU.

Este problema pode ser amenizado, até certo ponto. Uma solução é produzir “caches” entre o processador e a memória principal (arquitetura de Harvard), assim produzindo mais vias de troca de dados. Pode ser usado computação paralela (arquitetura NUMA), que adiciona mais hardwares para dividir os processamentos, aumentando a velocidade de processamento.

**2. Objetivo**

O objetivo desse relatório é analisar os resultados obtidos na criação de um processador com arquitetura Von-Neumann, bem como, relatar o processo de desenvolvimento deste, enfatizando as dificuldades encontradas e o aprendizado adquirido.

**3. Fundamentação Teórica**

A arquitetura interna de um processador baseado no modelo proposto por Von Neumann segue três premissas:

* As instruções e os dados são armazenados no mesmo espaço de memória
* Estas instruções são executadas de forma sequencial
* Cada espaço da memória é endereçado, o qual será necessário para referenciar a posição de determinado conteúdo

Entendendo está ideia base, infere-se que ao rearranjar os dados por comandados de instruções, é possível modificar o programa.

Assim, esta arquitetura é composta por (Figura 01):

* Memória
* UCP (Unidade Central de Processamento)
  + ULA (Unidade Lógica e Aritmética)
  + UC (Unidade de Controle)
  + Registradores
* E/S (Entrada/Saída)

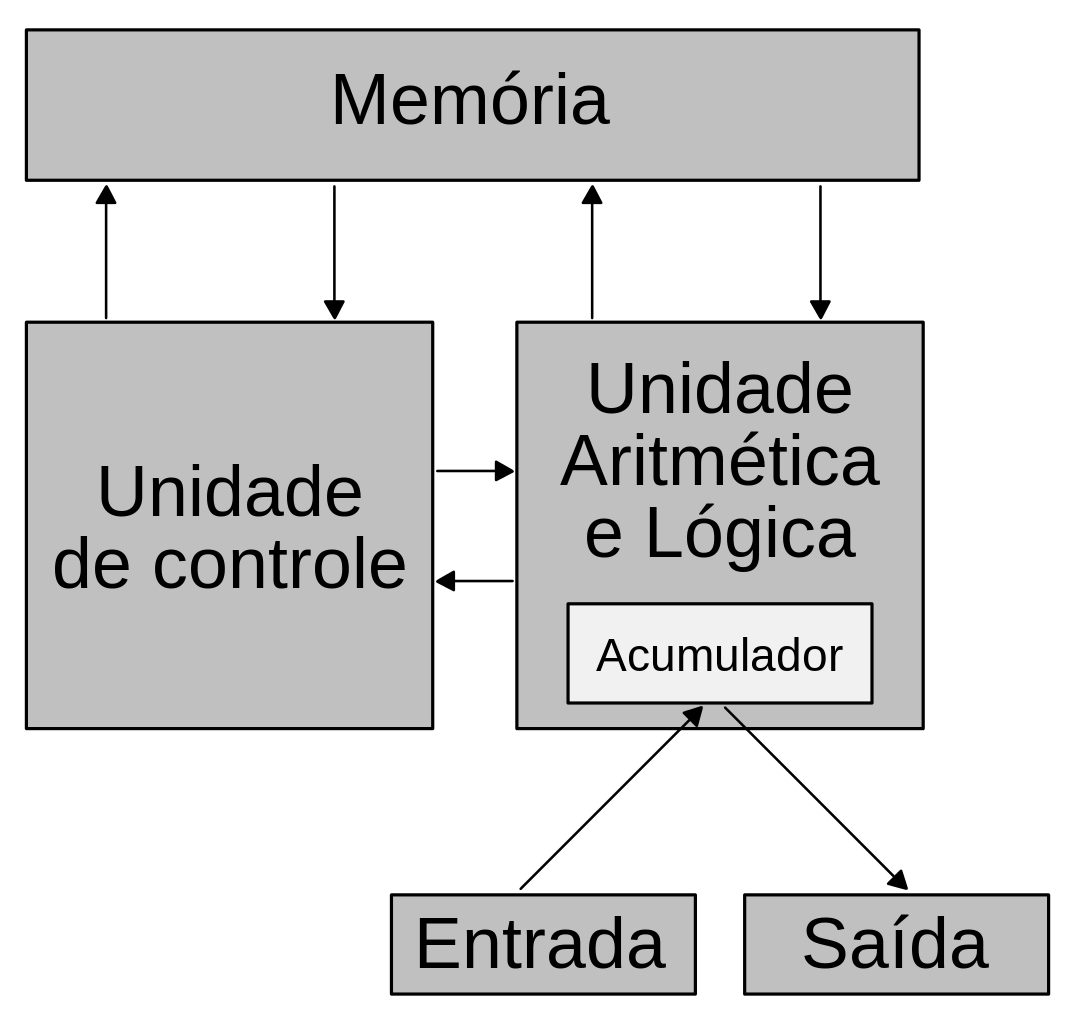


Figura 01: Arquitetura Von-Neumann

Neste modelo a UCP ou CPU (Central Processing Unit) possui a função de executar os programas armazenados na memória principal. Este, busca cada instrução, interpretando-a e logo após executando-a. Nela encontram-se a UC, que busca as instruções na memória principal e determina o tipo de cada instrução; a ULA, que realiza operações aritméticas necessárias para à execução das instruções; e um banco de registradores (memória cache), que é uma memória de alta velocidade, utilizada para armazenar dados (resultados) temporários e algumas informações necessárias para controle.

A partir dos dispositivos de entrada e saída é possível comunicar-se com a máquina. Assim, como o processador interpreta sinais binários, inviabiliza de um programador criar programas em código binário. Com isso, a linguagem mais baixo nível usada para esta comunicação é o Assembly.

Esta linguagem de montagem utiliza comandos mnemônicos (são símbolos que representam os códigos de máquina) para facilitar a interação programador-máquina.

**4.Processador**

O processador desenvolvido nesse projeto é composto por dois blocos principais, um bloco de controle, responsável pelo gerenciamento do dispositivo, coordenando os caminhos de dados por qual a informação processada irá passar. E um bloco de dados, nos quais são armazenadas as informações referentes à instrução e dados que serão processados posteriormente.

O bloco de controle é composto por uma máquina de estados que coordena a transição de etapas nas operações realizadas pelo processador. Esta recebe e envia sinais de comando a fim de habilitar processos de escrita, rota de dados através de multiplexadores e o controle de escrita da memória.

O bloco de dados é composto por seis registradores de uso específico, três registradores voltados ao armazenamento dos operadores que compõem a instrução, um registrador voltado à manutenção de informações para operações de movimentação memória-memória, um registrador-contador responsável pela contagem de instrução e apontamento de endereço de memória (Program Counter), e um registrador-contador responsável pelo apontamento de endereço de pilha(Stack Point Counter). Outros quatro registradores de uso geral são disponíveis para manipulação direta do usuário através dos comandos de movimentação de dados e operações aritméticas. Para realizar as conexões gerenciadas existem também uma série de multiplexadores, permitindo a comutatividade de conexões de dados e consequentemente um maior número de caminhos de dados dentro do processador, como citado anteriormente.

Os blocos de controle e dados, em conjunto, determinam a série de caminhos de dados nos quais as informações serão passadas. Estes caminhos de dados são determinados pelos OPCodes inseridos na instrução.

Este dispositivo apresenta um conjunto de 54 instruções dispostas em quatro categorias principais:

* Movimentação de dados;
* Operações lógicas e aritméticas;
* Operações de chamadas de função e manipulação de endereço;
* Operação de parada de processo;

As operações de movimentação de dados realizam o transporte de informação tanto a nível registrador-registrador quanto a nível registrador-memória. As operações registrador-registrador realizam a transferência de dados entre registradores distintos ou não. As operações de movimentação registrador memória permitem a alocação de dados advindos da memória para o registrador e vice versa, podendos estes serem alocados em endereços específicos ou em endereços pertencentes à pilha.

As operações aritméticas somente são realizadas a nível registrador-ULA-registrador. A ULA é composta por um conjunto de lógicas combinacionais voltadas à atender as necessidades operacionais do processador. Nelas são realizadas operações como soma, subtração, operações lógicas (And , Or , Not, Xor), shifting de dados existentes nos registradores e acréscimos ou decréscimos de um. As operações são selecionadas através de um código operacional gerado pelo bloco de controle, resultando em uma saída de dados referente à operação selecionada e um conjunto de “flags” utilizadas em operações de manipulação de endereço.

As operações de chamada de função e manipulação de endereços realizam os chamados saltos de instrução, fazendo com que o contador de programa aponte para um endereço diferente, determinado nos comandos Jump e Call, ignorando a contagem sequencial normalmente utilizada. As operações Jump podem utilizar as “flags” a fim de realizar os pulos condicionados, relativos à resultados de operações anteriores como comparações. A operação Call realiza a chamada de funções ou rotinas, armazenando o endereço de memória de próxima instrução em pilha e permitindo o regresso ao funcionamento normal após seu término.

A operação de parada de processo ou Halt constitui na prisão do processador em um determinado estado; geralmente utilizado ao fim de um conjunto de instruções.

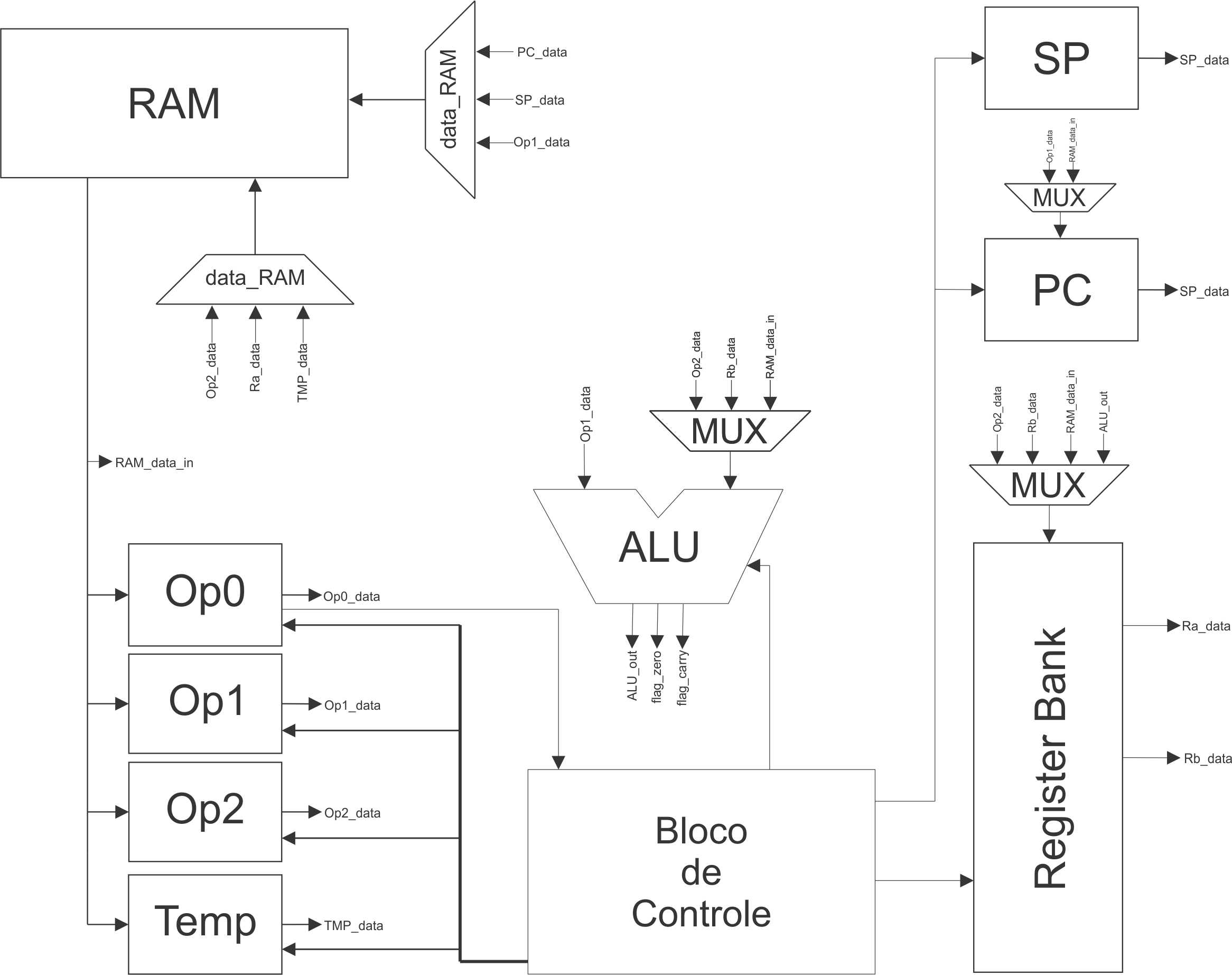


Figura 02: Esquemático do Processador

**5. Resultados**

Após simular o processador, foram obtidos resultados satisfatórios. As instruções implementadas, foram lidas, interpretadas e executadas corretamente. Assim, o funcionamento individual dos blocos e em conjuntos, foram interligados da maneira adequada para a arquitetura.

Algumas dificuldades foram encontradas no decorrer do projeto. Como, por exemplo, o grande número de funções a serem implementadas e algumas, como \*\*\* foram mais trabalhosas pelas suas complexidades de execução. Tivemos dificuldade, também, \*\*\*\*\*\*\*\*\*.

Com tudo, foi possível chegar ao resultado final com êxito. Mesmo que o programa não tenha sido otimizado, a maior parte deste foi escrito em baixo nível de VHDL. Com isso, as poucas funções prontas utilizadas no programa, permitiu uma maior segurança para controlar o passo-a-passo das execuções.

**6. Conclusão**

O objetivo desse projeto foi alcançado. Foi possível simular o comportamento de um processador com arquitetura Von-Neumann na linguagem VHDL. As instruções foram implementadas corretamente, entretanto, para fins de simplicidade, foram aplicadas instruções únicas para comandos similares.

Tal projeto serviu como um grande aprendizado, tanto para entender passo-a-passo o funcionamento de um processador, como para trabalhar a linguagem de programação utilizada.

Além do estudo sobre diferentes arquiteturas de processadores, o qual foi importante para entender a evolução de tais e o motivo de funcionarem assim. Com o estudo detalhado de cada parte do processo, foi notório a percepção das vantagens e desvantagens de utilizarmos (na maioria dos computadores de mesa atuais) essa arquitetura.

Posteriormente, serão realizados estudos na linguagem Assembly, para aprofundar mais o conhecimento sobre o funcionamento de microprocessadores e tentar otimizar seu desempenho, baseando-se nas necessidades impostas e os conhecimentos prévios.

**Referências**

GATTO, Elaine Cecília: Arquitetura de Jonh Von Neumann. Disponível em: <<https://www.embarcados.com.br/arquitetura-de-john-von-neumann/>>. Acesso em: 17 set. 2017.

D’AMORE, R.; VHDL: descrição e síntese de circuitos digitais. 2.ed. Rio de Janeiro: LTC, 2015. 292p.

STALLINGS, William. Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

VAHID, F.; Sistemas Digitais: projeto, otimização e HSLs; tradução Anatólio Laschuk. – Porto Alegre: Artmed, 2008. 560p.

WIKILIVROS: Introdução à Arquitetura de Computadores/Modelo de Von Neumann. Disponível em: <<https://pt.wikibooks.org/wiki/Introdu%C3%A7%C3%A3o_%C3%A0_Arquitetura_de_Computadores/Modelo_de_Von_Neumann>>. Acesso em: 17 set. 2017.

WIKIPÉDIA: Assembly. Disponível em: <<https://pt.wikipedia.org/wiki/Assembly>>. Acesso em: 17 set. 2017.