UIT - UNIVERSIDADE DE ITAÚNA

CIÊNCIA DA COMPUTAÇÃO

EDMILSON LINO CORDEIRO

DAVI VENTURA CARDOSO PERDIGÃO

JOÃO PAULO FERNADES ROCHA

Estrutura e Função do Processador

Pipeline

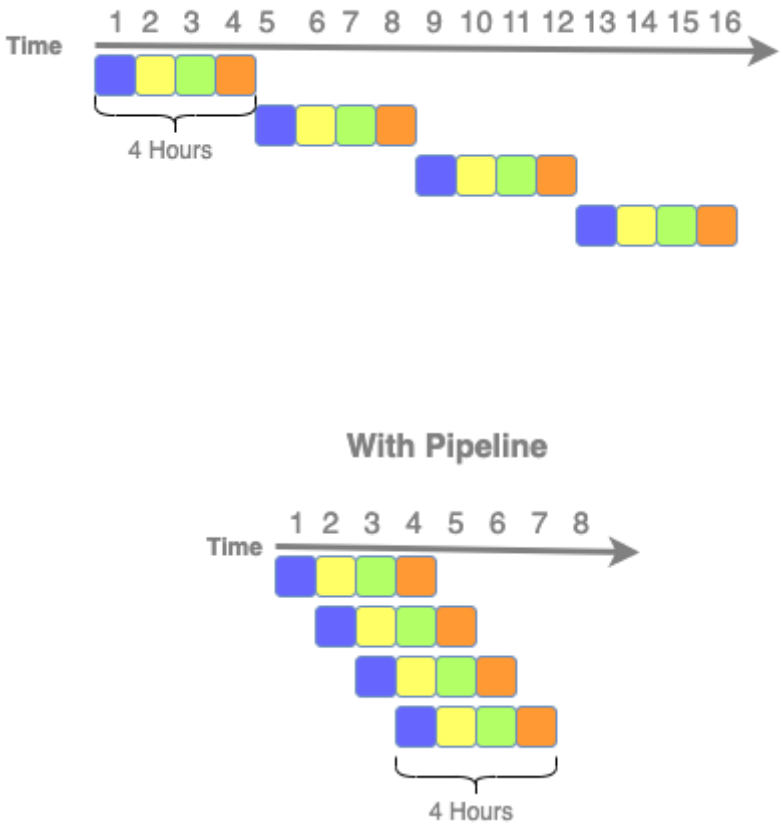
Arquitetura e Organização de Computadores II

Itaúna, 2021

Definição

Pipeline, muitas vezes traduzido para português como paralelismo, é uma técnica que permite os processadores executarem tarefas diferentes ao mesmo tempo. Essas tarefas são colocadas em uma fila de memória dentro do processador (CPU) onde aguardam o momento de serem executadas: assim que uma instrução termina o primeiro estágio e parte para o segundo, a próxima instrução já ocupa o primeiro estágio. Essa técnica aumenta o desempenho do processador e reduz o tempo de execução global de tarefas.

Figura 1 - Demonstrativo do tempo de execução
sem Pipeline e com Pipeline



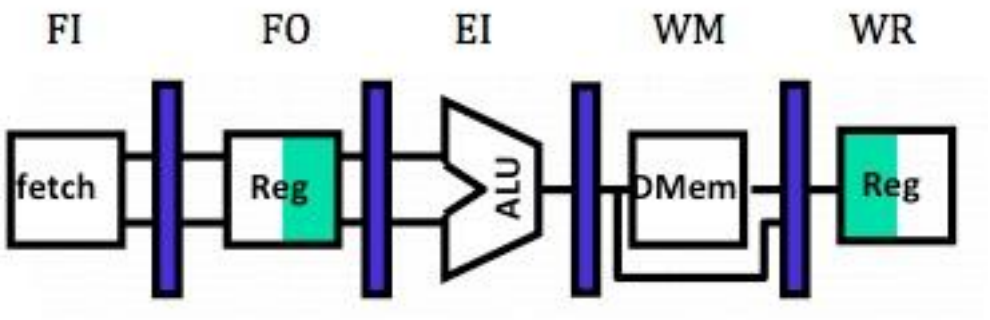
Fonte: PANTUZA, Gustavo. 2020.

Quando é carregada uma nova instrução, ela primeiramente passa pelo primeiro estágio, que trabalha nela durante apenas um ciclo de clock, passando-a para o estágio seguinte. A instrução continua então sendo processada sucessivamente pelo segundo, terceiro, quarto e quinto estágios do processador. A vantagem desta técnica, é que o primeiro estágio não precisa ficar esperando a instrução passar por todos os demais para carregar a próxima, e sim carregar uma nova instrução assim que se livra da primeira, ou seja, depois do primeiro pulso de clock. Porém, entre os problemas enfrentados estão a dependência de

instruções anteriores e desvios que dificultam o processo, bem como a diferença de complexidade de instruções que fazem com que as mesmas possam levar um tempo variável para execução.

Arquitetura Básica Pipeline em um Computador

Figura 2 - Processador adaptado para trabalhar com
Pipeline de cinco estágios



Fonte: BRITO, Alisson. Introdução a Arquitetura de Computadores.

A primeira mudança necessária é a separação da memória em duas partes independentes (ou duas memórias). Uma parte será utilizada apenas para instruções (na figura, Fetch), e outra apenas para os dados (DMem). Isso é necessário para que a etapa “FI” acesse a memória para buscar a próxima instrução, ao mesmo tempo em que a “WM” acessa a memória para salvar o resultado de outra instrução anterior. Se houvesse apenas uma memória para dados e instruções, isso não seria possível. Essa mudança vai contra o que foi projetado na Arquitetura de von Neumann, e considerado um grande avanço. Ela foi batizada de Arquitetura Harvard.

Outra mudança importante foi a adição de memórias intermediárias entre cada etapa. Na figura 2, essas memórias são representadas pelos retângulos preenchidos e sem nenhuma palavra sobre eles. Essas memórias são utilizadas para armazenar o resultado da etapa anterior e passá-lo para a etapa posterior no ciclo seguinte. Elas são necessárias porque as etapas não executam necessariamente sempre na mesma velocidade. Se uma etapa for concluída antes da etapa seguinte, seu resultado deve ser guardado nessas memórias para aguardar que a etapa seguinte conclua o que estava fazendo. Só então ela poderá receber o resultado da etapa anterior.

Visão geral do Pipeline

Em resumo, é o processo pelo qual uma instrução de processamento é subdividida em etapas, uma vez que cada uma destas etapas é executada por uma porção especializada da CPU, podendo colocar mais de uma instrução em execução simultânea. Isto traz um uso mais racional da capacidade computacional com ganho substancial de velocidade. A técnica de segmentação de instruções é utilizada para acelerar a velocidade de operação da CPU, uma vez que a próxima instrução a ser executada está normalmente armazenada nos registradores da CPU e não precisa ser buscada da memória principal que é muito mais lenta.

- Analogia com uma “Lavanderia Doméstica”

Como exemplo, vamos pegar uma lavanderia, e supondo que cada uma das seguintes etapas possa ser realizada em 30 minutos:

1) Colocar a roupa na máquina de lavar

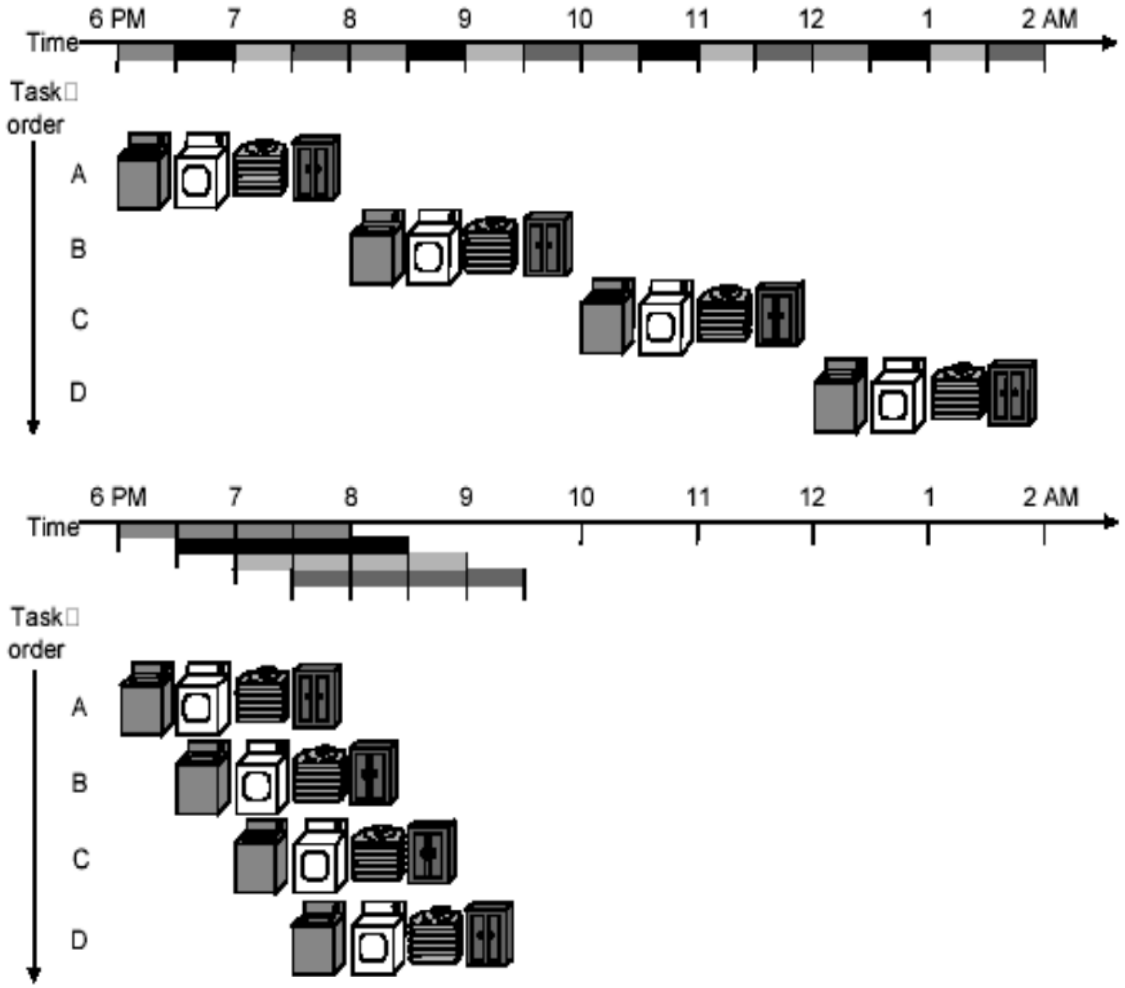
2) Depois de lavada, colocá-la na máquina de secar roupa

3) Depois de seca, passar a ferro

4) Depois de passada, arrumá-la no armário

Supondo-se que cada uma destas etapas leve 30 minutos para ser realizada, a lavagem de um cesto de roupas continuará levando 2 horas para ser realizada. Entretanto, podemos iniciar a lavagem de um cesto de roupas a cada 30 minutos, até que tenhamos 4 cestos sendo lavados simultaneamente, um em cada etapa do “pipeline”. Depois das primeiras 2 horas, teremos um cesto de roupa lavada a cada 30 minutos. Ao final do dia teremos lavado muito mais cestos de roupa do que sem o uso de pipeline.

Figura 3 - Analogia com lavanderia, com e sem Pipeline



Fonte: SILVA, Gabriel. Arquitetura de Computadores II.

Princípio Básico

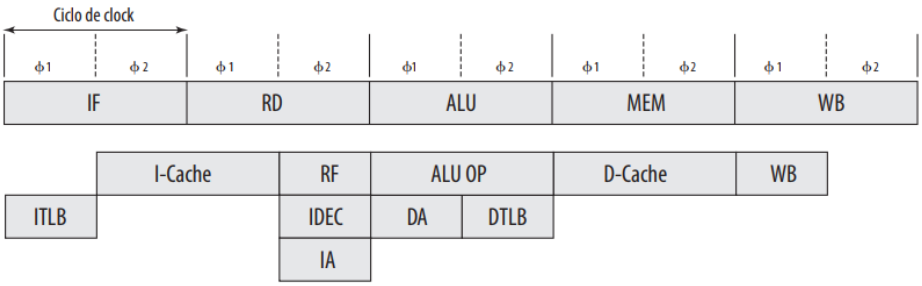
Por mais que tenhamos um clock baseado na instrução mais lenta, o pipeline consegue ter um tempo de execução por instrução menor. Na verdade, cada instrução irá demorar o tempo de clock, no entanto o fato de várias instruções estarem sendo executadas em paralelo faz com que a vazão de instruções seja maior.

Vamos considerar um exemplo em que tem 5000 instruções a serem executadas. Suponhamos um clock de 10 nano-segundos. Um processador não pipeline levaria 5000 * 10 = 50000 ns. Se tivermos um processador com um pipeline de 12 estágios temos que ao final da execução de todas instruções seria 50000 / 12 = 3333,3 ns.

Instruções do MIPS

Como já foi estudado, os processadores são componentes desenvolvidos metodicamente e seguem sempre um padrão de funcionamento e realização das tarefas, portanto na arquitetura MIPS tal fato não seria diferente. Em relação ao Pipeline, no MIPS, o processo de leitura, execução e atualização das instruções é dividido em cinco etapas:

Figura 4 - Divisão em cinco etapas



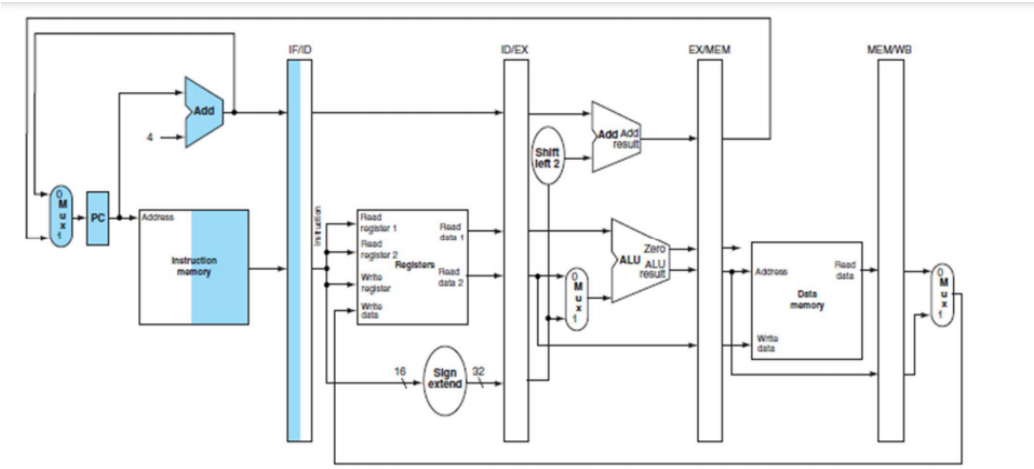
IF = busca da instrução
RD = leitura
MEM = acesso à memória
WB = atualizar
I-Cache = acesso à cache de instruções
RF = busca do operando do registrador
D-Cache = acesso à cache de dados
ITLB = tradução endereço da instrução
IDEC = decodificação da instrução
IA = calcular endereço da instrução
DA = calcular endereço virtual de dados
DTLB = traduzir endereço de dados
TC = verificar rótulo de cache de dados

Fonte: STALLINGS, William. Pag. 417. 2009.

1° Etapa: IF

Responsável pela busca da instrução, traduzir seu endereço virtual para o endereço físico e enviá-lo ao endereço em que a instrução está armazenada.

Figura 5 - Exemplo Etapa IF

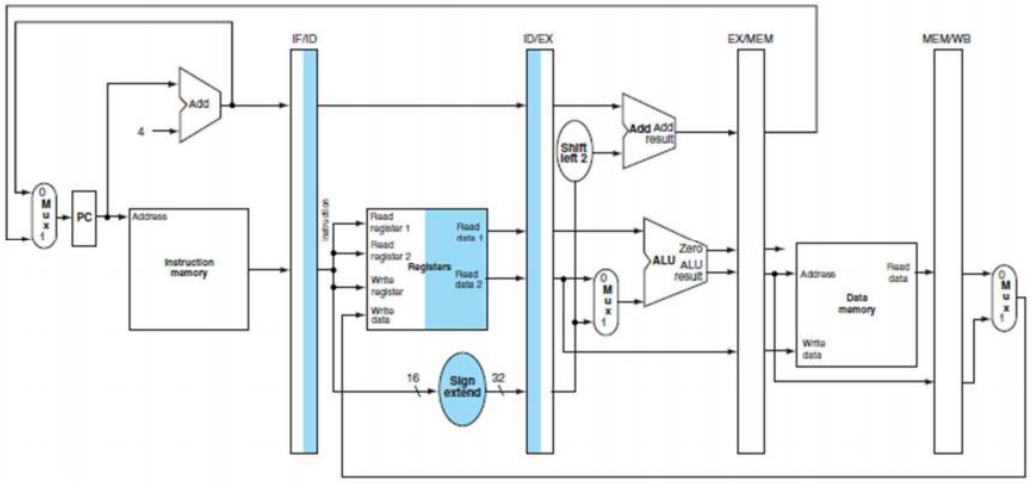


Fonte: UFF (Universidade Federal Fluminense), Cap. 5. 2017

2° Etapa: RD

Nessa etapa que ocorre a decodificação da instrução, a busca do operando que será necessário para sua execução, a leitura do banco de registradores e, caso a instrução a ser executada seja um desvio, será realizado o cálculo do endereço que ele será armazenado.

Figura 6 - Exemplo Etapa RD



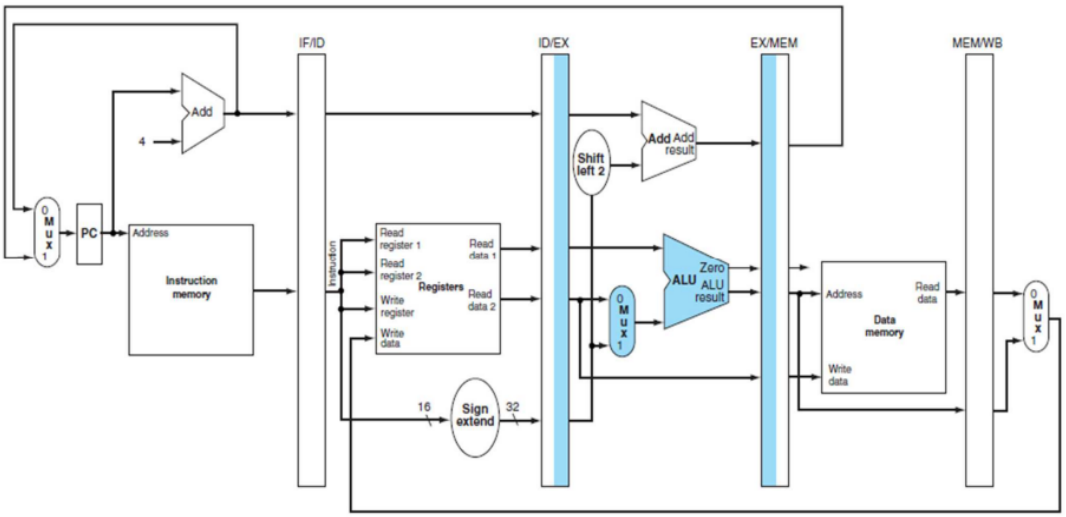
Fonte: UFF (Universidade Federal Fluminense), Cap. 5. 2017

3° Etapa: ALU

Nesta etapa que as instruções são efetivamente executadas, conforme a natureza da instrução, será realizado umas das seguintes ações.

- Se a instrução for de natureza registrador-registrador, a ULA (Unidade Lógica e Aritmética) executará a tarefa.
- Se for uma instrução de leitura ou escrita, o endereço virtual de dados é calculado.
- Se a instrução é um desvio, suas condições são verificadas e será decidido se o desvio será ou não tomado.

Figura 7 - Exemplo Etapa ALU

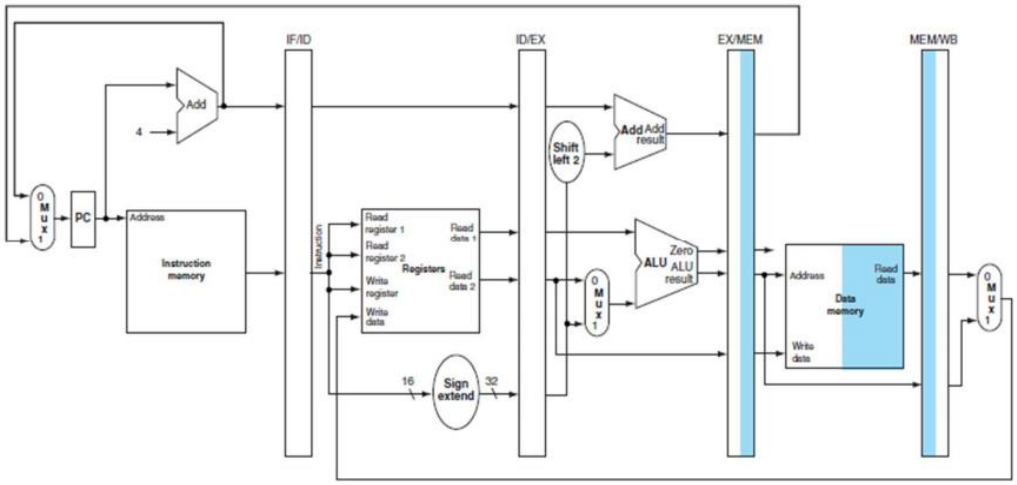


Fonte: UFF (Universidade Federal Fluminense), Cap. 5. 2017

4° Etapa: MEM

Etapa onde a memória cache é acessada para o envido do endereço físico resultante da execução da instrução.

Figura 8 - Exemplo Etapa MEM

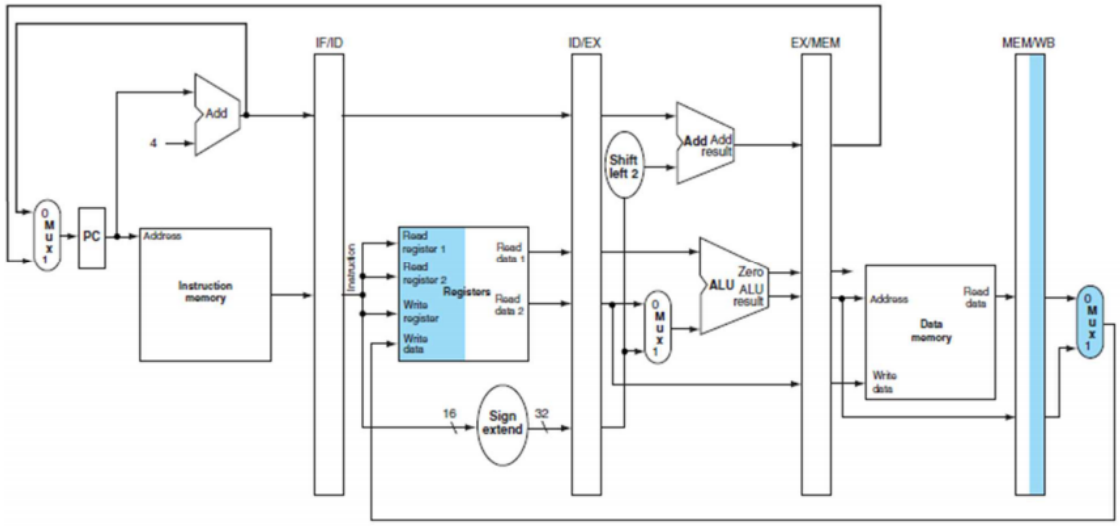


Fonte: UFF (Universidade Federal Fluminense), Cap. 5. 2017

5° Etapa: WB

Processo onde o resultado da instrução executada é armazenado nos registradores.

Figura 9 - Exemplo Etapa WB



Fonte: UFF (Universidade Federal Fluminense), Cap. 5. 2017

Conclusão

Apesar do ganho de desempenho, uma arquitetura pipeline com seis estágios apresenta problemas que não ocorrem em uma arquitetura com dois estágios. Isto porque, quanto maior o número de estágios, maior a dependência de um estágio em relação ao outro. Isso é apenas um exemplo para concluirmos que, o uso de uma arquitetura pipeline traz ganhos de performance para a máquina, porém é uma técnica que requer um projeto cuidadoso (por parte dos projetistas de hardware, do processador) para que possa alcançar resultados ótimos em termos de desempenho.

Referências Bibliográficas

- PANTUZA, Gustavo. **Organização e Arquitetura de Computadores – Pipeline em Processadores**. Disponível em: <https://blog.pantuza.com/artigos/organizacao-e-arquitetura-de-computadores-pipeline-em-processadores>. Acesso em: 12 de março de 2021.
- SÉRGIO, Luiz. **Arquitetura e Organização de Computadores**. Disponível em: < https://educapes.capes.gov.br/bitstream/capes/206151/2/apostila%20de%20AOC_Luiz%20S%C3%A9rgio.pdf>. Acesso em: 01 de março de 2021.
- SILVA, Gabriel. **Arquitetura de Computadores II – Pipeline**. Disponível em: <https://dcc.ufrj.br/~gabriel/arqcomp2/Pipeline.pdf>. Acesso em: 11 de março de 2021.
- STALLINGS, William. **Arquitetura e Organização de Computadores**. Tradução da 8a edição. Editora Prentice Hall Brasil, 2002.
- TANENBAUM, Andrew S. **Organização Estruturada de Computadores**. Tradução da 5a edição. Editora Prentice Hall Brasil, 2007.
- UFF - Universidade Federal Fluminense. **Capítulo 5 Pipeline.** Disponível em: <http://www.professores.uff.br/lbertini/wp-content/uploads/sites/108/2017/08/Cap-5-Pipeline.pdf>. Acesso em: 10 de março de 2021.
- VASCONCELOS, Laércio. **Hardware Total**. 1ª Edição. Editora Makron Books, 2002