**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS**

ENGENHARIA DE COMPUTAÇÃO

ARQUITETURA DE COMPUTADORES

Erick Matheus Lopes Pacheco – 18711630

Hiago Silva Fernandes – 18726455

Marcos Antônio Junior Vasconcellos – 18720920

Victor Reis – 18726471

**HVEM 2.0 – PROCESSADOR PIPELINE EM VHDL**

DOCUMENTAÇÃO, DETALHAMENTO E RESULTADOS DO PROJETO

CAMPINAS

Novembro 2019

SUMÁRIO

[1. INTRODUÇÃO 3](#_Toc24734414)

[1.1. TOPOLOGIA DA CPU 3](#_Toc24734415)

[2. ESPECIFICAÇÃO 7](#_Toc24734416)

[2.1. REGISTRADORES 7](#_Toc24734417)

[2.1.1. ENDEREÇO DOS REGISTRADORES 7](#_Toc24734418)

[2.2. FORMATO DAS INSTRUÇÕES (OPCODE) 8](#_Toc24734419)

[2.3. UNIDADE DE CONTROLE 11](#_Toc24734420)

[3. RESULTADOS 14](#_Toc24734421)

[3.1. TESTES REALIZADOS 14](#_Toc24734422)

[3.2. RESULTADOS E DISCUSSÃO 16](#_Toc24734423)

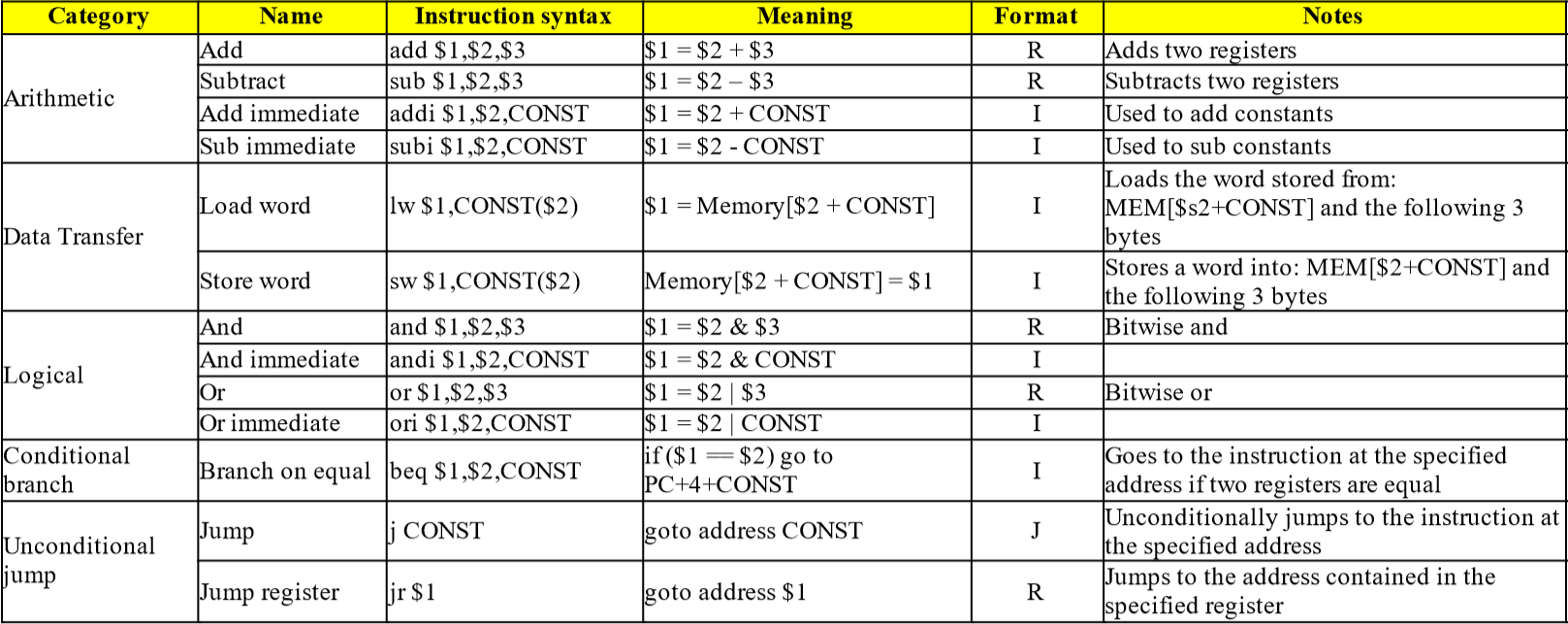
[4. BIBLIOGRAFIA 17](#_Toc24734424)

# INTRODUÇÃO

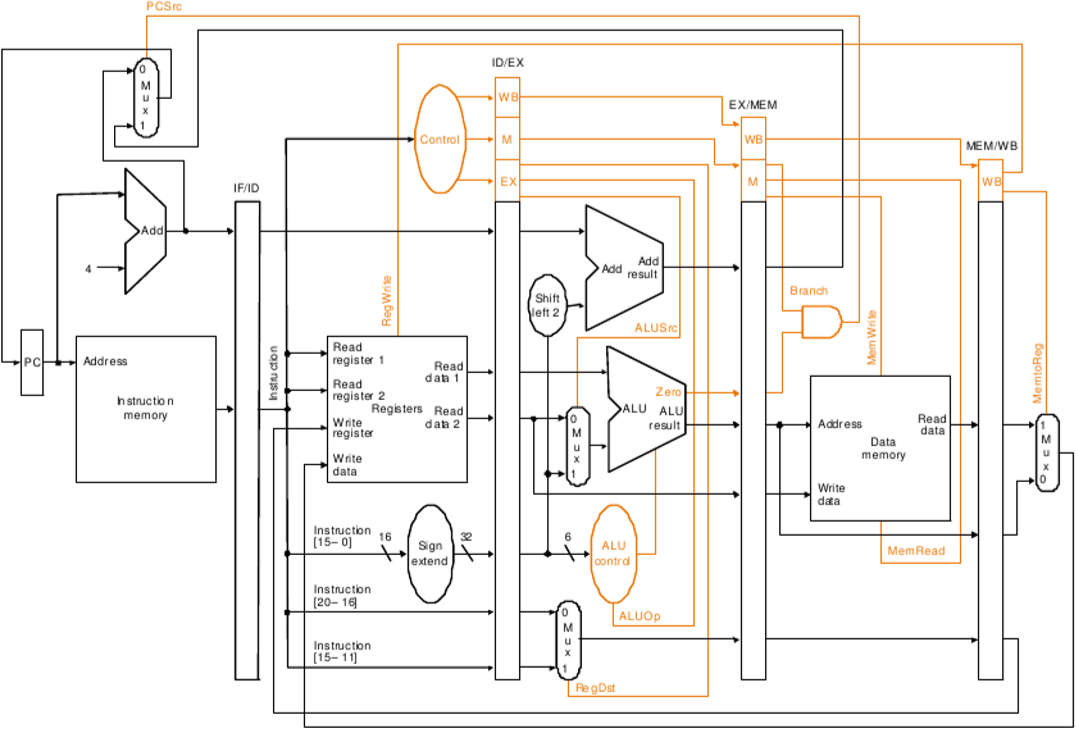
Dá-se a proposta de projeto pelo professor: projetar e desenvolver um processador pipeline em VHDL, que seja capaz de realizar operações aritmética, leitura e escrita de instruções, jump e branch, com e sem valores imediatos atribuídos.

Desse modo, o grupo se organizou de maneira a todos trabalharem juntos, sem que houvesse falhas de comunicação ou conflito de atividades. Para isso utilizamos cinco ferramentas: **Discord** para comunicação VOIP do grupo, **Google Docs** e **Draw.io** para projetar a CPU, a máquina de estados e definir suas instruções; **Visual Studio Code** com o plugin Live Share para programação simultânea em VHDL, além do **Intel Quartus Prime** para a compilação e simulação do nosso código.

## TOPOLOGIA DA CPU

Foi proposto um processador capaz de executar as seguintes instruções:

Em detrimento ao projeto anterior, onde batizamos nossa CPU de “HVEM” (representando as iniciais dos membros integrantes do grupo), batizar-se-á com o mesmo nome, como sendo uma versão *melhorada* da CPU anterior

Devido ao paralelismo exigido, torna-se necessário a divisão do processador em estágios, com registradores capazes de armazenar as instruções e seus sinais para cada um desses estágios, de modo a não perder as informações. Seguindo essa linha de pensamento, nos foi fornecido o seguinte datapath base:

Devido ao fato de que o datapath base não executar as instruções de jump incondicional, mais a frente estaremos mostrando o datapath completo e adaptado para a implementação de todas as instruções propostas.

O processador operará com instruções e dados de 32 bits, ou seja, possuirá a capacidade de armazenar até 232 (4.294.967.296‬) valores, o que torna possível a plena execução das instruções requisitadas, bem como a possibilidade de uma expansão de projeto para implementar mais instruções e armazenar valores maiores, caso desejado.

Na página seguinte, uma imagem esquematizada do datapath completo da CPU pipeline, com a capacidade de executar todas as instruções.

* Em **vermelho**, tudo referente a Unidade de Controle;
* Em **laranja**, os fios que estabelecem comunicação entre os componentes;

# ESPECIFICAÇÃO

# DESENVOLVIMENTO

## RESULTADOS

# CONCLUSÕES

# BIBLIOGRAFIA

PANNAIN, Ricardo. “**CAPÍTULO 4 – O PROCESSADOR**”; Google Drive. Disponível em <https://drive.google.com/file/d/1EWK2zLVzMs2ON1PAyg\_bJHlp-w9Ce9mC/view?usp=sharing>. Acesso em novembro de 2019.

“**HOW TO CONVERT 8 BITS TO 16 BITS IN VHDL?**”, Stack Overflow. Disponível em <https://stackoverflow.com/q/17451492>. Acesso em novembro de 2019.

ANEXO I