Centro Federal de Educação Tecnológica de Minas Gerais

Introdução ao Projeto Tomasulo (Trabalho III)

Aluno: Lucas Henrique Ferreira Sousa

1.1) Introdução:

O presente trabalho aborda a implementação de um algoritmo de Tomasulo sem especulação. Por meio de uma estrutura de tópicos, será realizada uma descrição dos elementos do algoritmo e de alguns de seus componentes:

O projeto é composto por quatro unidades funcionais: duas para operações com inteiros, sendo uma destinada à (soma, subtração e comparação) e outra para (multiplicação/divisão) ; e duas para operações em ponto flutuante, com uma dedicada à (soma e subtração e comparação), e outra à (multiplicação e divisão). É importante destacar que, conforme as recomendações do PDF do Trabalho III, será incluída uma unidade funcional EXTRA para cálculo de endereço haja visto que foi permitida e recomendada.

As unidades funcionais (FUs) são descritas:

Tipo da FU	**Ciclos em EX**	**Quantidade de FU's**	**Espaços na Estação de Reserva**
Inteiros (Soma/Subtração /Comparações)	1	1	3
Inteiros (Mul/Div)	5	1	4
Ponto Flutuante (Soma/Subtração)	4	1	3
Ponto Flutuante (Mul/Div)	15	1	4
Cálculo de Endereço	1	1	4 Store 4 Load

As Unidades de Inteiros oferecem suporte a operações que exigem comparações, como SLTU, BEQ e BNE. Essas operações necessitam de comparações para verificar se um valor é menor que outro, se são iguais, entre outras condições.

Agora que as unidades funcionais foram apresentadas, podemos explorar o conjunto de instruções utilizado neste projeto, que inclui:

1.2) Conjunto de instruções:

Instrução	Descrição	FU utilizada
DADDIU	Soma Inteiro imediato	FU inteiros
ADD.D	Soma Inteiro ou Soma Ponto Flutuante	FU inteiros ou FU Ponto Flutuante *(Dependendo do tipo de registrador)*
SUB.D	Subtração Inteiro imediato ou Subtração Ponto Flutuante	FU inteiros ou FU Ponto Flutuante *(Dependendo do tipo de registrador)*
MUL.D	Multiplicação Inteiro ou Multiplicação Ponto Flutuante	FU inteiros ou FU Ponto Flutuante *(Dependendo do tipo de registrador)*
DIV.D	Divisão inteiros ou Divisão ponto flutuante	FU inteiros ou FU Ponto Flutuante *(Dependendo do tipo de registrador)*
LOAD	Leitura da memória	FU cálculo de endereço
STORE	Armazenar na memória	FU cálculo de endereço
BEQ	Branch if Equal (Desvio se igual)	FU inteiros

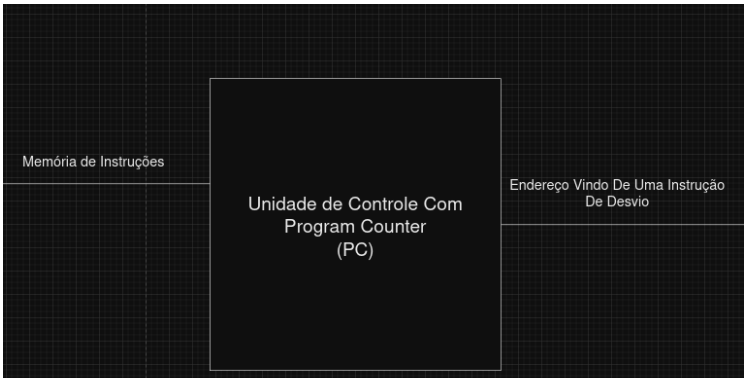
BNE	Branch if Not Equal (Desvio se diferente)	FU inteiros
DSLT	Menor que	FU inteiros

1.3) Estágios do Tomasulo:

IF: Nesse estágio, as instruções armazenadas na memória de instruções são trazidas para a fila de instruções. Esse processo interage diretamente com o Program Counter (PC), que contém o endereço da próxima instrução a ser buscada na memória e enviada para a fila de instruções.
ID/IS: Nesse estágio, as instruções que vem da fila de instrução são decodificadas e despachadas para suas respectivas estações de reserva. É importante destacar que, nesta etapa, são verificadas possíveis condições de hazard, como o hazard estrutural causado da lotação das unidades de reserva.
EX: EX é o estágio de execução das instruções. Neste estágio, para desvios, o Program Counter (PC) é atualizado com base na condicional atrelada (bne ou beq); para operações de load e store, o endereço de memória é calculado; e para as demais instruções, é realizado o cálculo do resultado a ser armazenado no registrador de destino.
MEM: No estágio de acesso à memória, as instruções de store acessam a memória para salvar um determinado valor, enquanto as instruções de load acessam a memória para ler um valor e transferi-lo para um registrador de destino.
WB: O estágio de Write-Back (WB) é responsável por atualizar o banco de registradores, enviando os dados através do Common Data Bus (CDB). O CDB é um barramento que conecta diversos componentes, assim como no algoritmo original de Tomasulo, permitindo o bypassing para as estações de reserva e outros componentes. (Na pasta deste projeto, há uma imagem com o diagrama do processador, detalhando essas conexões.)

1.4) Componentes do Projeto de Hardware Tomasulo:

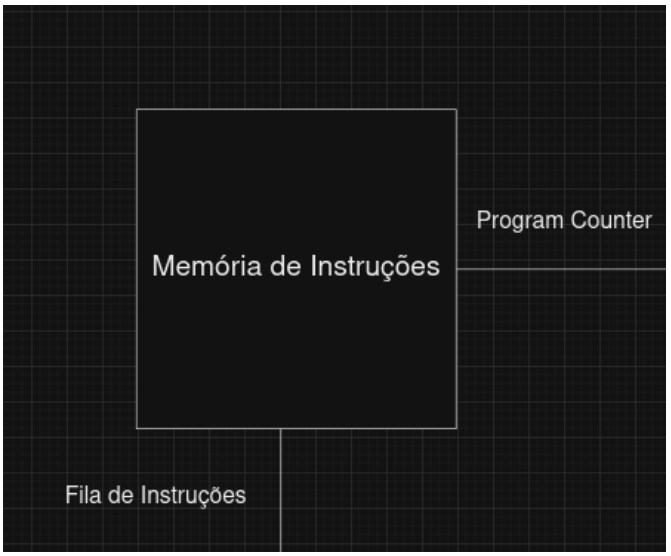
1.4.1) Program Counter (Com unidade de Controle inclusa):



Basicamente, o Program Counter (PC) está conectado tanto à memória

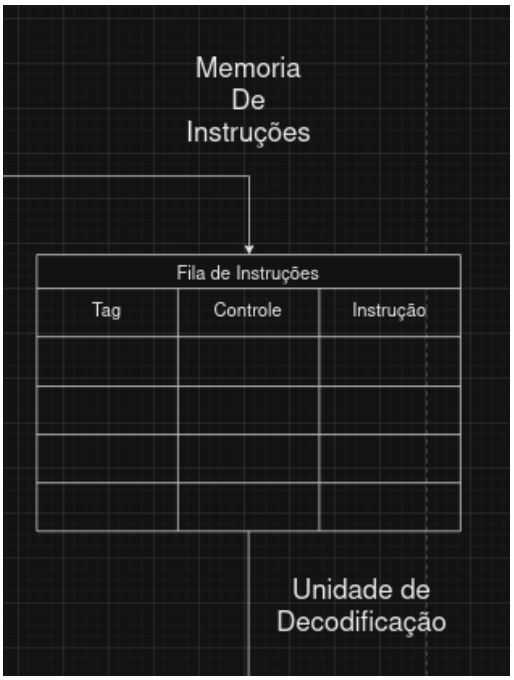
de instruções quanto a uma unidade de controle adicional que recebe o resultado da Unidade Lógica e Aritmética (ULA), utilizada por BNE e BEQ, junto com um sinal que autoriza a atualização do PC (Para fins de simplificação foi indicado como Endereço Vindo De uma instrução de Desvio na imagem). Isso garante que, no estágio de execução (EX), o endereço do PC seja atualizado corretamente caso o desvio seja tomado. A conexão com a memória de instruções permite que ela receba o endereço do PC e envie a instrução correspondente para a fila de instruções de forma adequada.

1.4.2) Memória de Instruções:



A memória de instruções armazena as instruções do programa. Ao receber o endereço fornecido pelo Program Counter (PC), ela permite a continuidade do fluxo do programa, enviando as instruções em ordem para a fila de instruções.

1.4.3) Memória de Instruções:

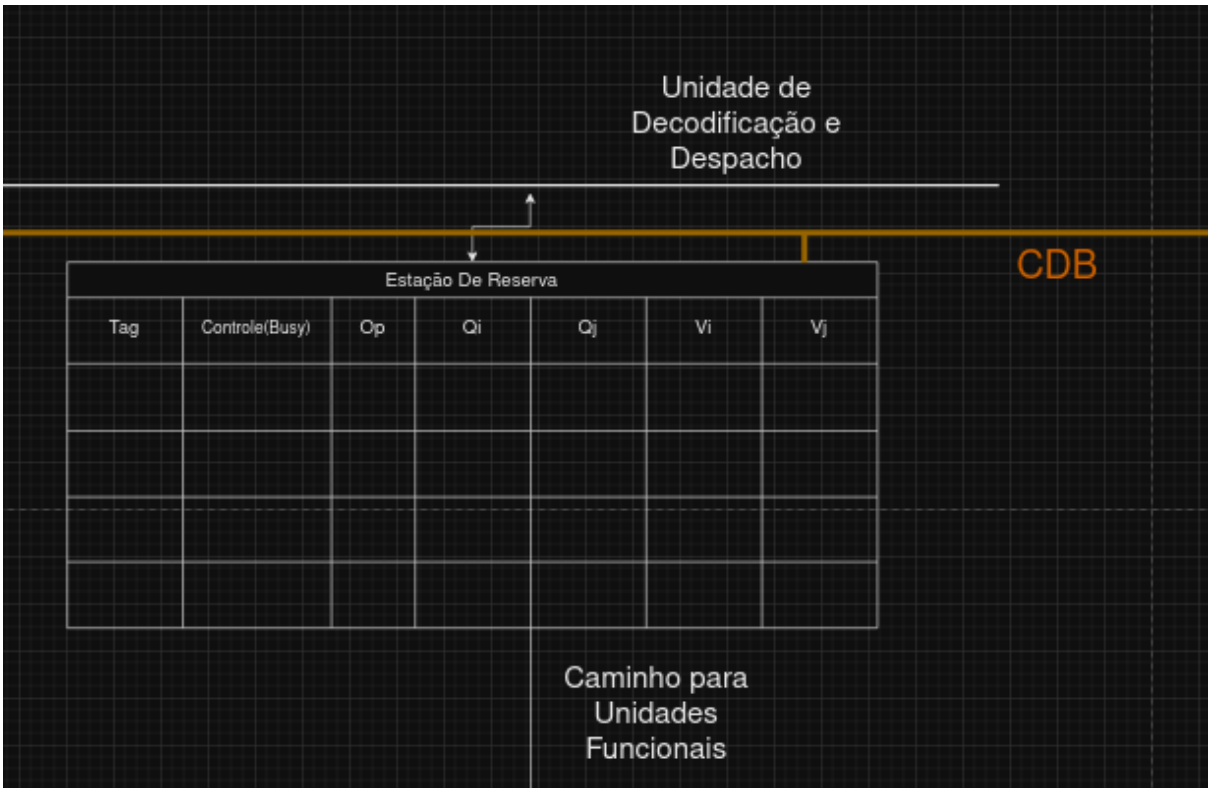


A fila de instruções é um buffer que utiliza TAGs para endereçamento e contém um sinal de controle junto com a instrução. As instruções são despachadas em ordem; após o despacho, a instrução é removida da fila, e o sinal de controle indica que o campo está vago, permitindo que uma nova instrução ocupe o espaço.

1.4.4) Unidade de Decodificação e Despacho:

A unidade de decodificação traduz a instrução para um formato que as estações de reserva e unidades funcionais possam processar. Nesse estágio, ela também verifica eventuais hazards estruturais relacionados às estações de reserva, uma vez que está diretamente conectada à FIFO. Se necessário, a unidade pode interromper o andamento da FIFO, atrasando o despacho para evitar conflitos.

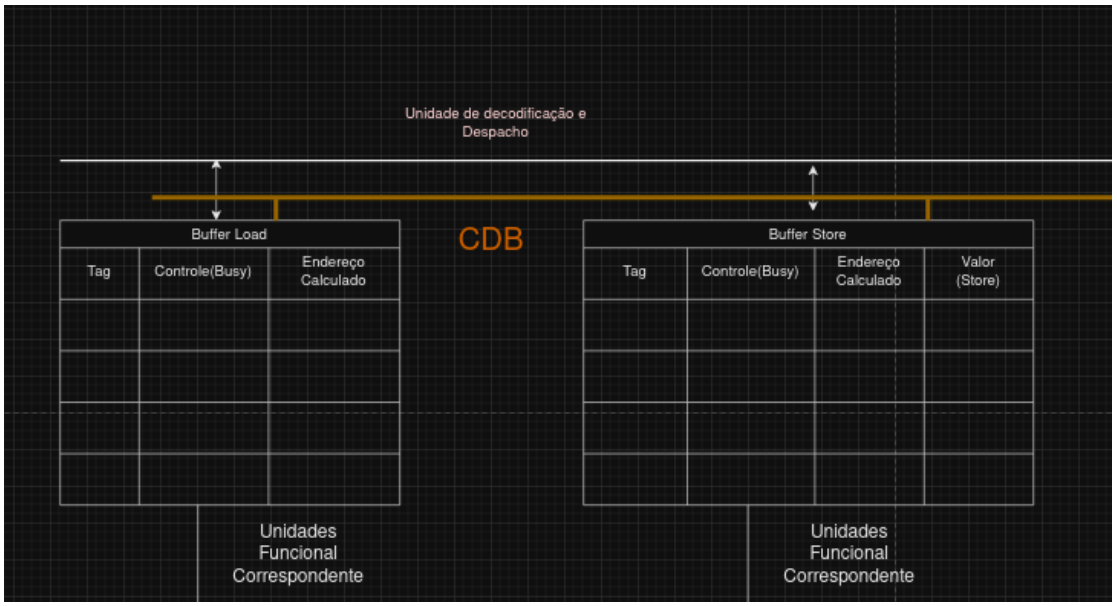
1.4.4) Estações de Reserva:



As estações de reserva neste projeto de hardware seguem a mesma especificação do algoritmo de Tomasulo discutido em sala de aula. Elas recebem as instruções das unidades de despacho e decodificação e as encaminham para as unidades funcionais correspondentes. No diagrama do

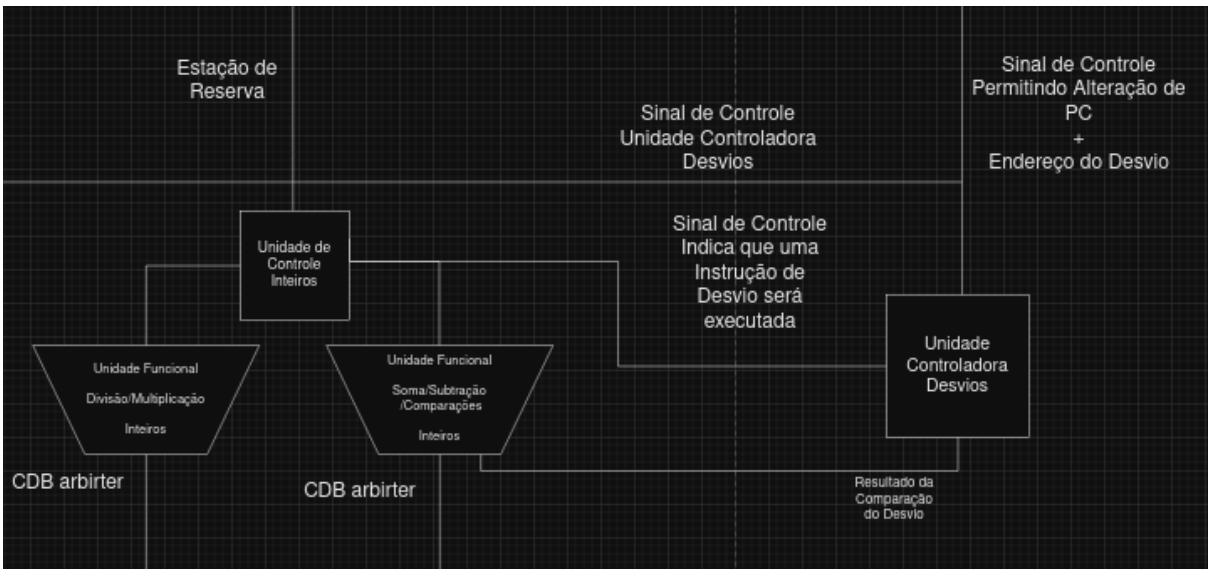
projeto (disponível na pasta do projeto), haverá uma distinção clara entre as estações de reserva: uma será destinada para operações com inteiros e outra para operações com ponto flutuante, com as respectivas indicações explícitas.

1.4.5) Buffers Load e Store :



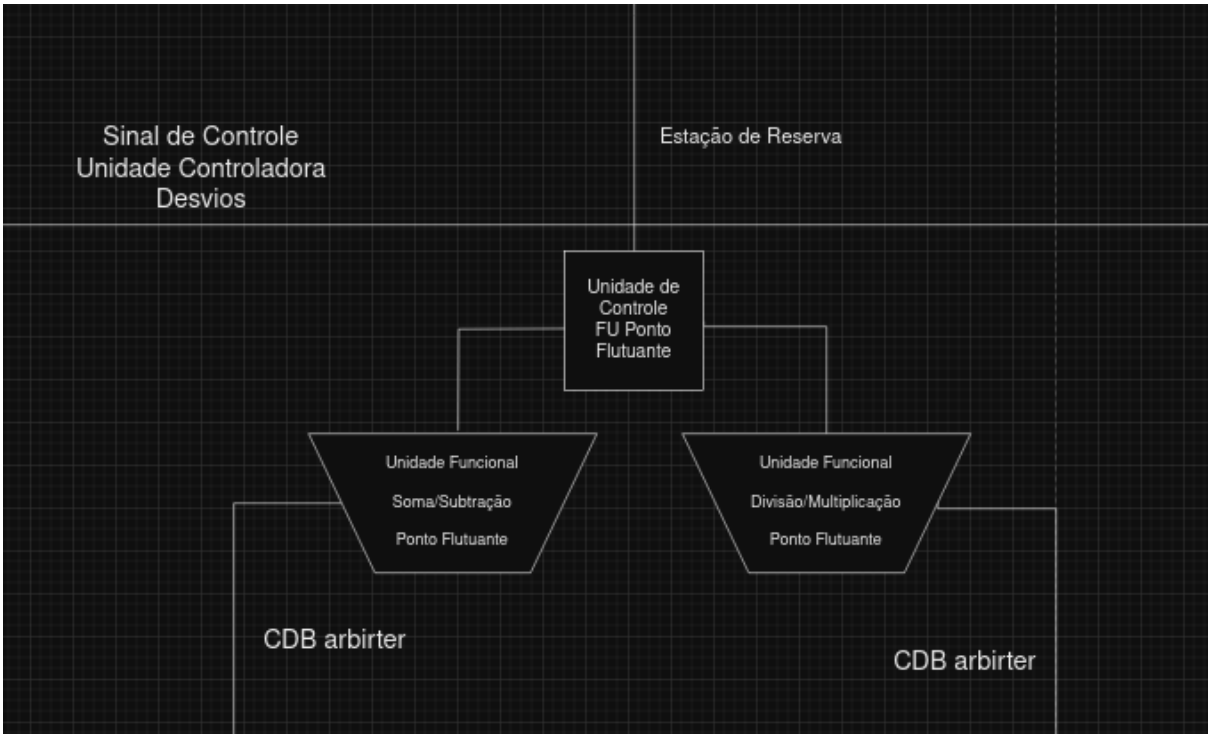
Buffers atuam como se fossem unidades de reserva; como discutido em sala de aula, a abordagem é essencialmente a mesma. No entanto, eles têm uma função específica para operações de load e store. Para buffers de load, a principal tarefa é armazenar endereços e dados que estão esperando para ser lidos da memória, enquanto para buffers de store, a função é armazenar endereços e dados que devem ser gravados na memória.

1.4.6) Unidades funcionais e controle de inteiros + Unidade Controladora de Desvio:



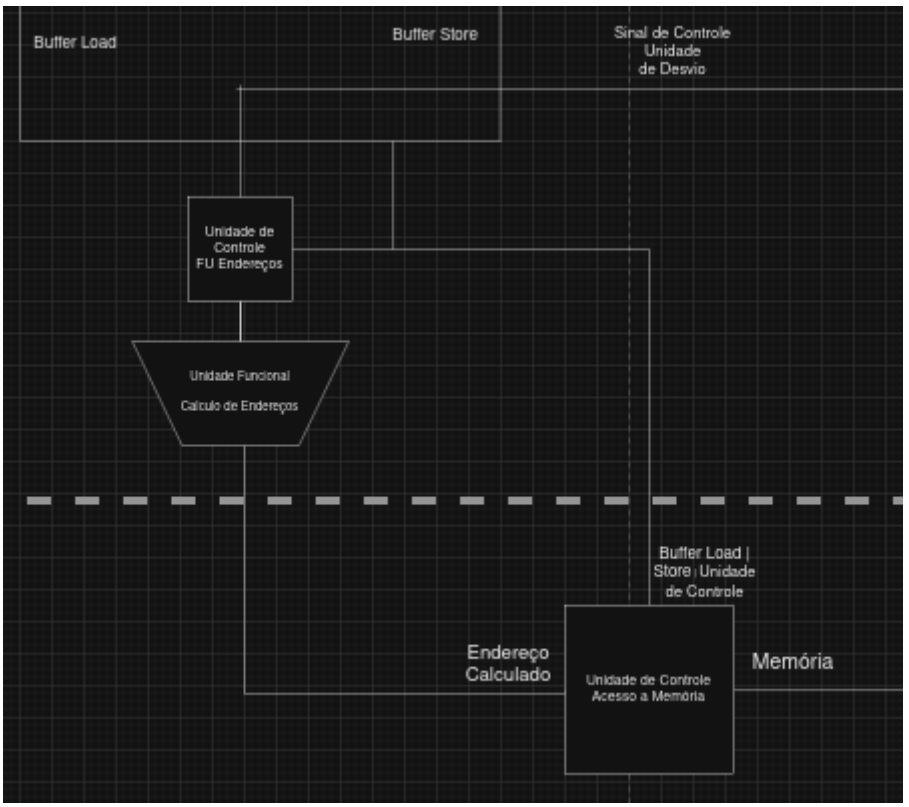
As unidades funcionais são alimentadas pelas operações enviadas pelas estações de reserva. Uma unidade controladora determina a instrução correta para cada unidade funcional e pode atrasar o estágio de execução, se necessário. Além disso, ela fornece dados relevantes para a unidade de controle de desvios. Quando uma instrução de desvio é identificada, a unidade controladora do desvio, em comunicação com a unidade de controle de inteiros, reconhece que as demais instruções em um ciclo superior a execução da instrução de desvio devem aguardar a conclusão da instrução de desvio. Ela então emite um sinal para as outras unidades funcionais, instruindo-as a esperar (uma vez que este é um Tomasulo sem especulação) é necessário esperar o desvio. Caso o desvio seja tomado, o endereço de desvio é fornecido ao PC para sua atualização.

1.4.7) Unidades funcionais e controle de Ponto Flutuante :



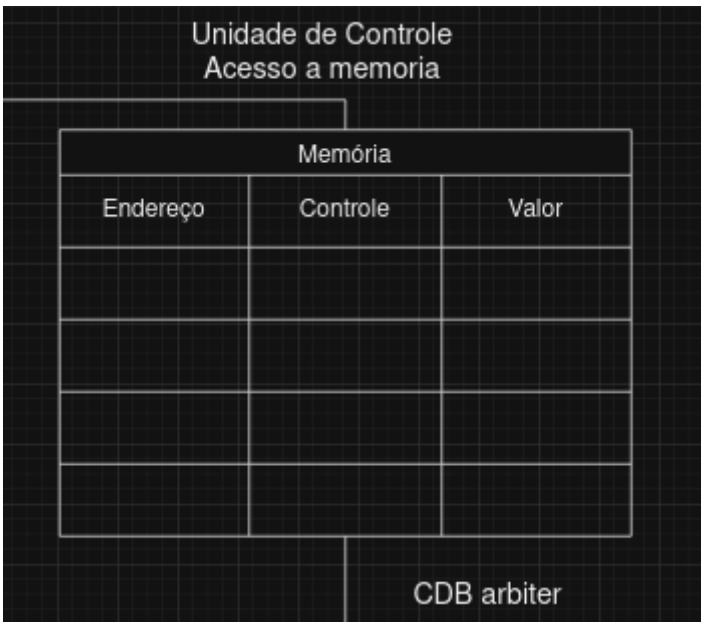
As unidades funcionais são controladas pela Unidade de Controle de Ponto Flutuante, que designa a operação correta a ser executada e pode atrasar o estágio de execução, se necessário. Como mencionado anteriormente, trata-se de um algoritmo Tomasulo sem especulação, portanto, é essencial ter cuidado com desvios. A unidade de controle também recebe sinais da unidade de desvios para garantir que as instruções aguardem a resolução do desvio, quando necessário. o dado é enviado para o CDB arbirter unidade que gerencia o que é salvo no CDB.

1.4.8) Unidades funcionais e controle relacionadas a memória (Buffers load e store e Unidade Funcional Calculo de Memória) :



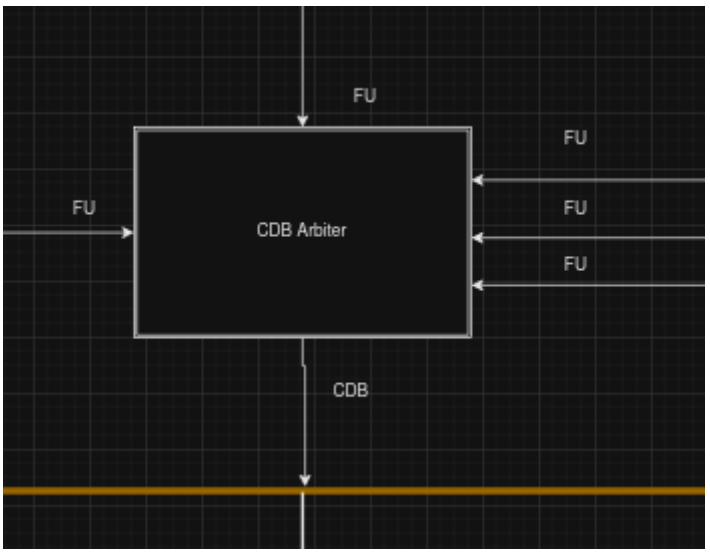
A Unidade de Controle de FU Endereços é responsável por gerenciar a operação da unidade funcional de inteiros. Assim como as outras unidades, ela recebe um sinal de controle da Unidade de Desvio, que indica se um desvio está sendo executado, para que a unidade possa aguardar sua conclusão, se necessário. A unidade funcional (FU) também interage com outra Unidade de Controle responsável pelo acesso à memória. Esta última está conectada aos buffers, e ao receber um endereço e um valor prontos para armazenamento (store), a unidade encaminha os dados necessários para que sejam gravados na memória. Para operações de leitura (load), a unidade de controle fornece o dado lido diretamente da memória.

1.4.9) Memória



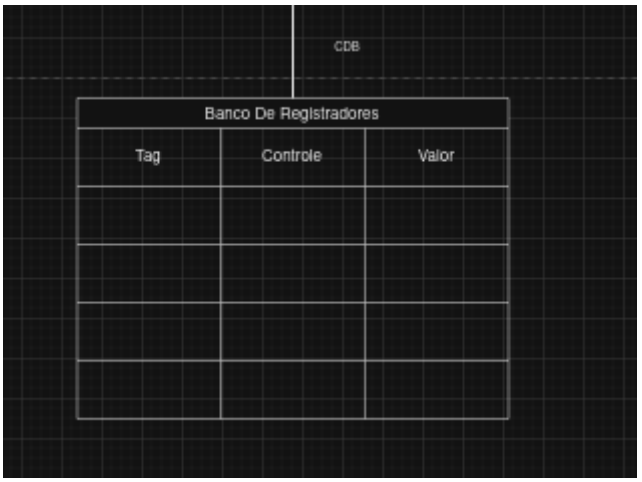
A memória está conectada à Unidade de Controle de Acesso à Memória e ao CDB arbiter. A Unidade de Controle de Acesso à Memória gerencia o acesso à memória, controlando tanto os dados que serão armazenados quanto os endereços associados, uma vez que ela está em contato com os buffers ela envia também o dado a ser salvo na memória no caso de um store. No caso de uma operação de leitura (load), ela determina o endereço de onde o dado será obtido para envio ao CDB. A memória possui endereços e valores associados, além de sinais de controle que indicam se um determinado endereço é válido.

1.4.10) CDB arbiter:



O árbitro do CDB (CDB arbiter) determina qual Unidade Funcional (FU) terá permissão para escrever no CDB, garantindo que apenas uma unidade funcional escreva por vez. Isso é necessário para evitar conflitos, já que não é permitido que duas unidades funcionais escrevam simultaneamente no CDB.

1.4.11) Banco de Registradores :



Dado um conjunto de registradores inteiros e de ponto flutuante, cada um identificado por suas respectivas tags, o CDB (Common Data Bus) é capaz de fornecer os dados diretamente desses registradores. Além disso, o CDB pode também gravar novos valores no banco de registradores, garantindo que as operações de leitura e escrita sejam realizadas de forma eficiente e sincronizada.

2) Conclusão:

Esta introdução teve como objetivo apresentar os conceitos e componentes do algoritmo de Tomasulo que serão implementados, além de esclarecer possíveis dúvidas sobre o diagrama do projeto, que está disponível na pasta de projetos. É importante destacar que, no diagrama, alguns sinais de controle foram omitidos para evitar poluição visual. No entanto, todos esses sinais estão detalhados neste PDF. Conceitos adicionais serão abordados na seção dedicada à descrição da lógica de funcionamento de cada componente utilizado.