# Recuperação de Projeto

Arquitetura PicoMIPS – Hierarquia de Memória

PCS3412 – Organização e Arquitetura de Computadores

Luciana da Costa Marques

Nusp 8987826

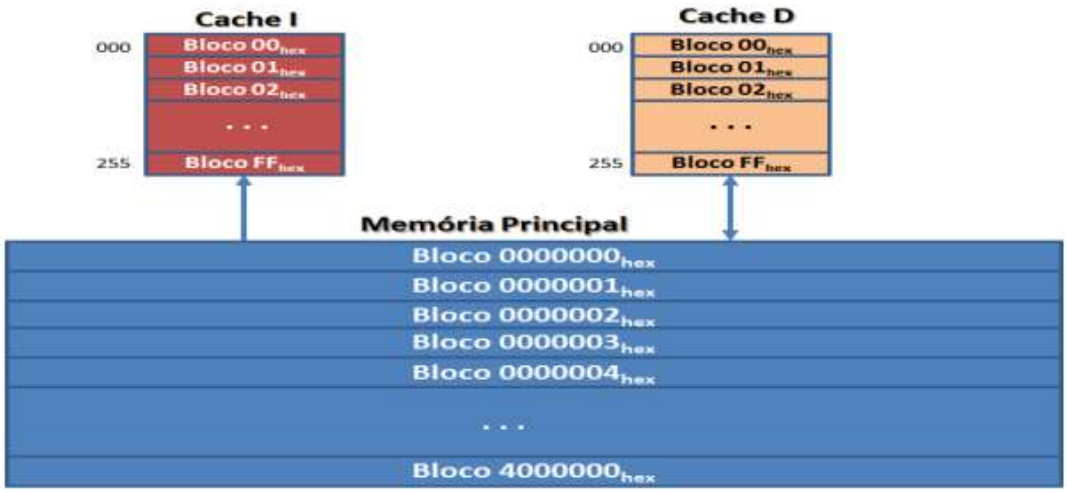
1. INTRODUÇÃO

Esta documentação visa descrever o desenvolvimento do projeto de recuperação da Hierarquia de Memória do processador PicoMips. Tal projeto levou em conta as especificações do projeto original, tais como tempos de acesso e os tamanho das memórias, a saber:

- Cache de instrução
  - Tempo de acesso: 5ns (tanto leitura quanto escrita);
  - 256 blocos de 16 palavras;
  - Mapeamento direto.
- Cache de dados
  - Tempo de acesso: 5ns (tanto leitura quanto escrita);
  - 128 posições com dois blocos cada de 16 palavras;
  - Mapeamento associativo de dois blocos.
- Memória Principal
  - Tempo de acesso: 100 ns (tanto leitura quanto escrita);
  - 14 bits de endereço;
  - Uma palavra por endereço.

Considerou-se que o endereço é composto por 16 bits (a separação dos campos para cada caso é melhor descrita nos capítulos seguintes) e que uma palavra é formada por 32 bits (isto é, 4 bytes).

Figura 1



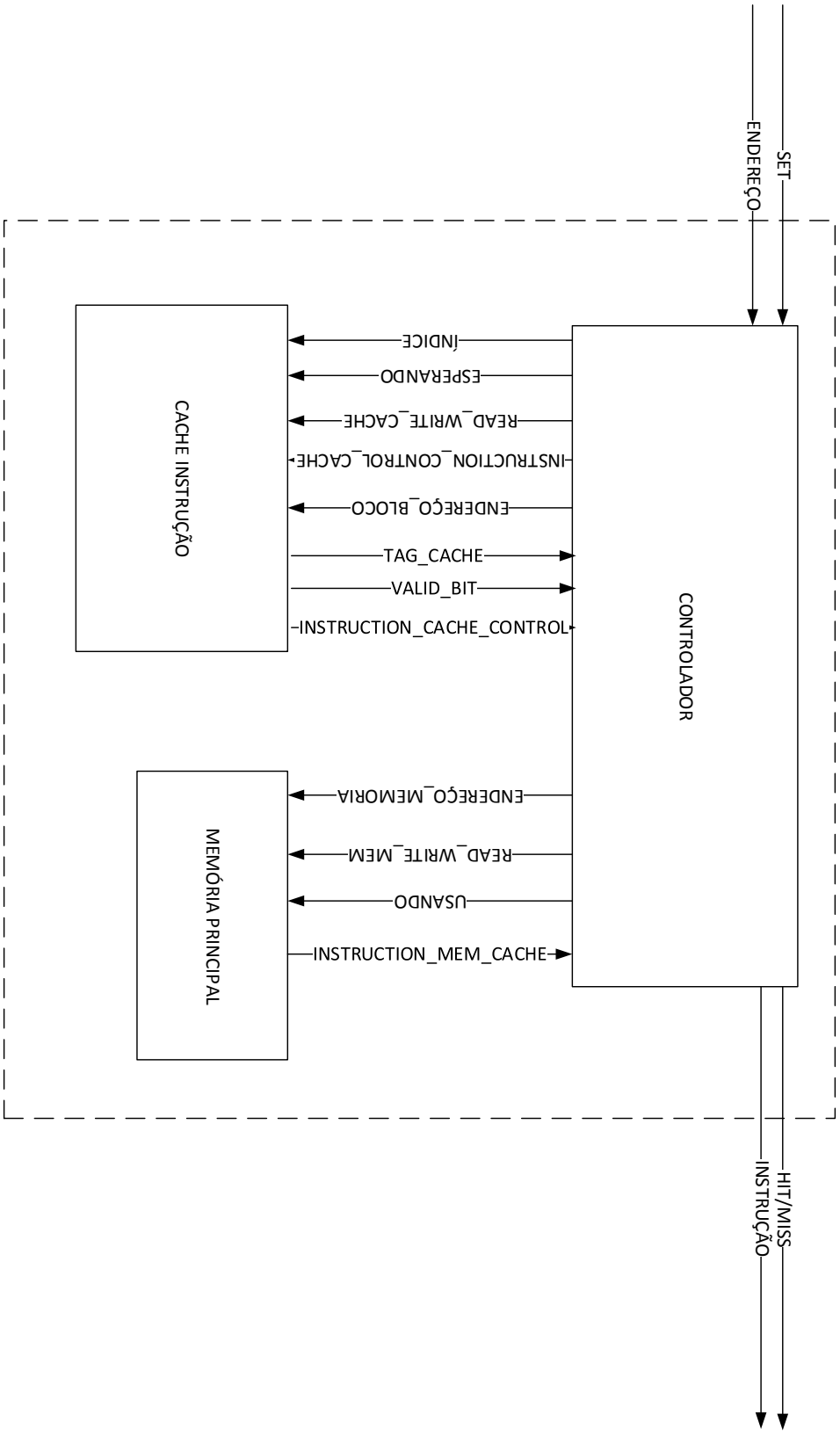
## 2. PROJETO DA HIERARQUIA DE MEMÓRIA

A Hierarquia de memória, por praticidade de implementação, foi dividida em duas unidades: a do cache de instruções e a do cache de dados.

### 2.1. CACHE DE INSTRUÇÃO

A hierarquia para o cache de Instrução foi feita na maneira da figura 2. Ela é composta por três módulos: i) a memória principal; ii) o cache de instruções e iii) o controlador. Optou-se nesse projeto por não haver comunicação direta entre o cache e a memória, tendo portanto o controlador como intermediário para a comunicação.

Figura 2


ENDEREÇO
SET
CACHE INSTRUÇÃO
ÍNDICE
ESPERANDO
READ_WRITE_CACHE
INSTRUCTION_CONTROL_CACHE
ENDEREÇO_BLOCO
TAG_CACHE
VALID_BIT
INSTRUCTION_CACHE_CONTROL
CONTROLADOR
MEMÓRIA PRINCIPAL
ENDEREÇO_MEMORIA
READ_WRITE_MEM
USANDO
INSTRUCTION_MEM_CACHE
INSTRUÇÃO
HIT/MISS

### 2.1.1.DESCRIÇÃO DOS MÓDULOS INTERIORES

#### 2.1.1.1. Cache de Instrução

O cache de instrução tem como entrada um sinal de Read/Write, o índice que se deseja acessar, o endereço da palavra buscada dentro do bloco, e um sinal para envio de instrução caso se deseje escrever no cache. Há ainda um sinal de "esperando", que é usado pelo controlador para indicar se ocorreu ou não um miss, melhor explicado no capítulo específico do controlador, adiante. Os sinais de saída são os respectivos bit de validade e tag associados ao índice de entrada, e um sinal de saída da instrução lida.

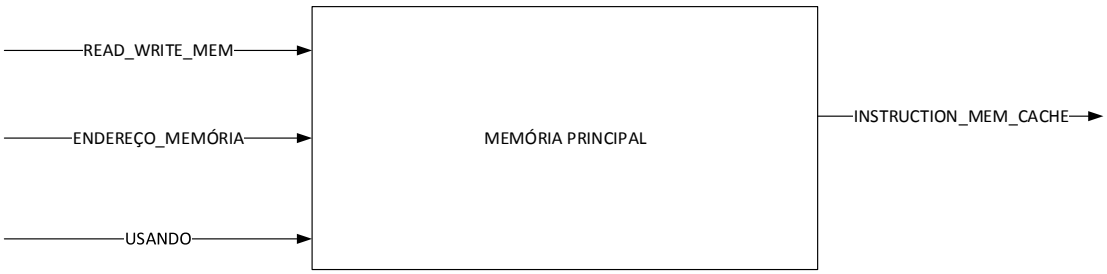
Em termos de código, o cache foi implementado basicamente como um array de vetores de 32 bits (tamanho das instruções). Há ainda um array para os bits de tag e outro para os bits de validade.

Figura 3: Cache de Instrução


#### 2.1.1.2. Memória Principal

A memória principal consiste em um array de 2**14 posições, sendo que cada posição tem 32 bits. Os seus sinais de entrada consistem em um sinal de R/W, o endereço que se deseja acessar e um sinal de saída de instrução lida.

Figura 4: Memória Principal




#### 2.1.1.3. Controlador

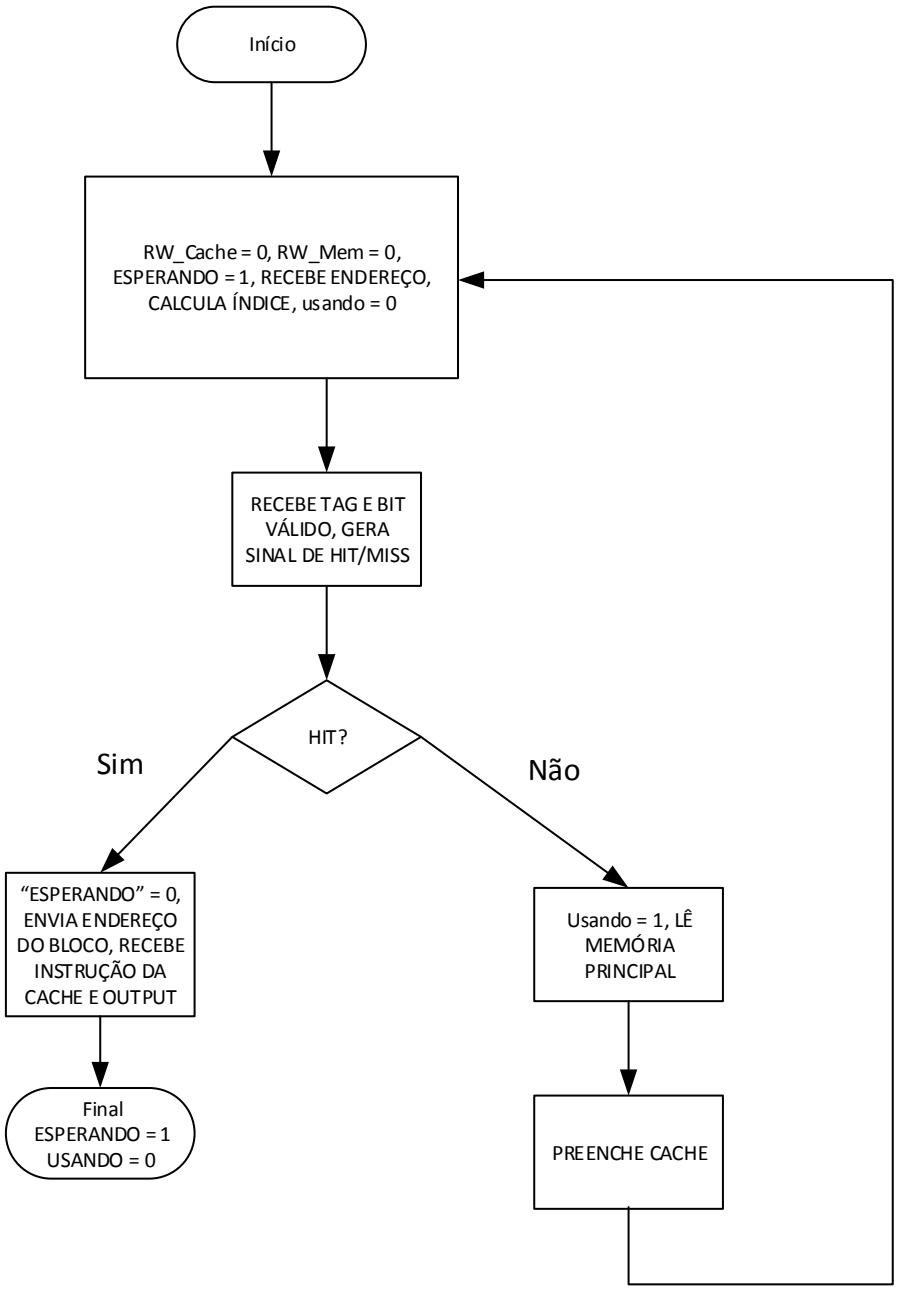
O projeto do Controlador seguiu a ASM da figura 5. Existem dois sinais indicando estados: ESPERANDO e USANDO.

Os estados possíveis são:

- ESPERANDO = 1 E USANDO = 0: estado inicial por default, indica que o sistema está esperando receber um hit;

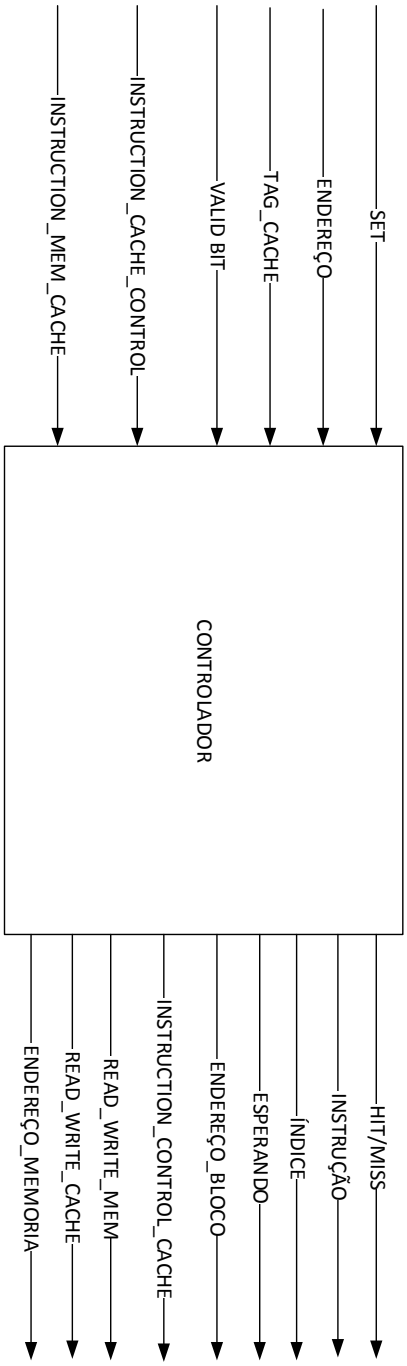
- ESPERANDO = 1 E USANDO = 1: estado em que ocorreu um miss e se está buscando a instrução específica na memória;
- ESPERANDO = 0 E USANDO = 0: estado em ocorreu um hit (não se está mais “esperando”) e preenche-se a cache.

Figura 5: ASM do controlador




Além da ASM, o projeto do controlador também segue o diagrama de blocos da figura 6. Como entrada há um sinal de clock “set”, o endereço a ser lido, a tag e o bit de validade enviados pela cache, a instrução obtida pela cache e a instrução obtida pela memória. Como saída há os sinais de hit/miss e de instrução lida (quando ocorre um hit), os sinais de índice, tag e de localização da palavra dentro do bloco (enviados para cache), bem como os sinais de Read/Write do cache e da memória, a instrução a ser escrita na cache e o endereço enviado à memória.

Figura 6: Diagrama de blocos do controlador

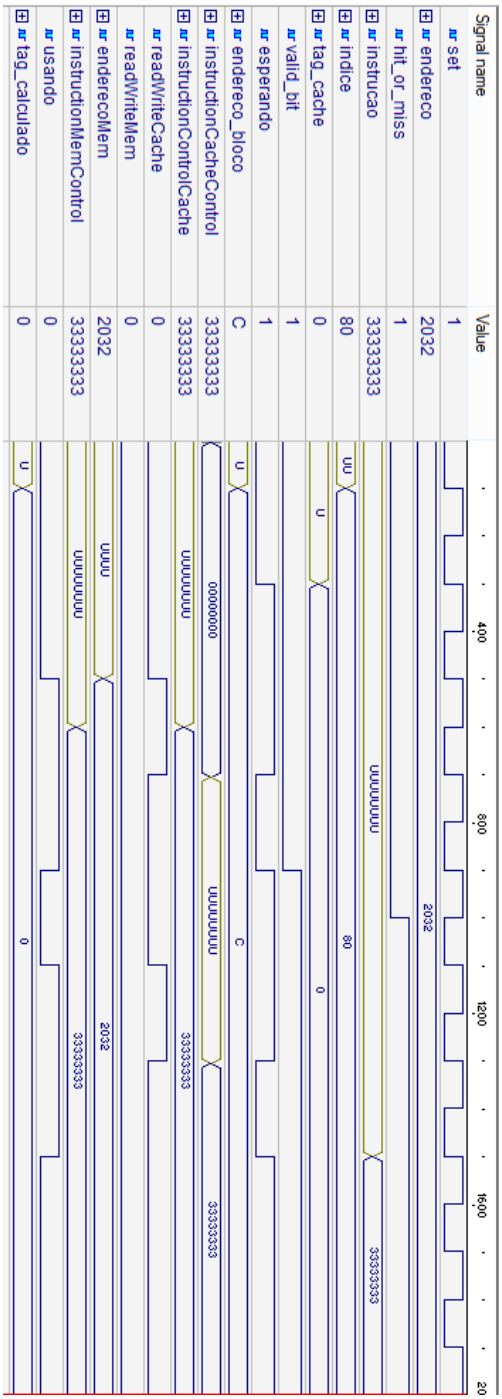



O endereço recebido pelo controlador é interpretado da seguinte maneira:

- Bits 0-1: bits de offset, não utilizados;
- Bits 2-5: bits de localização de 16 palavras;
- Bits 6-13: índice calculado (8 bits – 256 posições);
- Bits 14-15: bits de tag.

2.1.2.SIMULAÇÃO

Preencheu-se na memória principal, no endereço 8242 (0010000000110010) o valor (00110011001100110011001100110011). Inicialmente o cache está vazio.

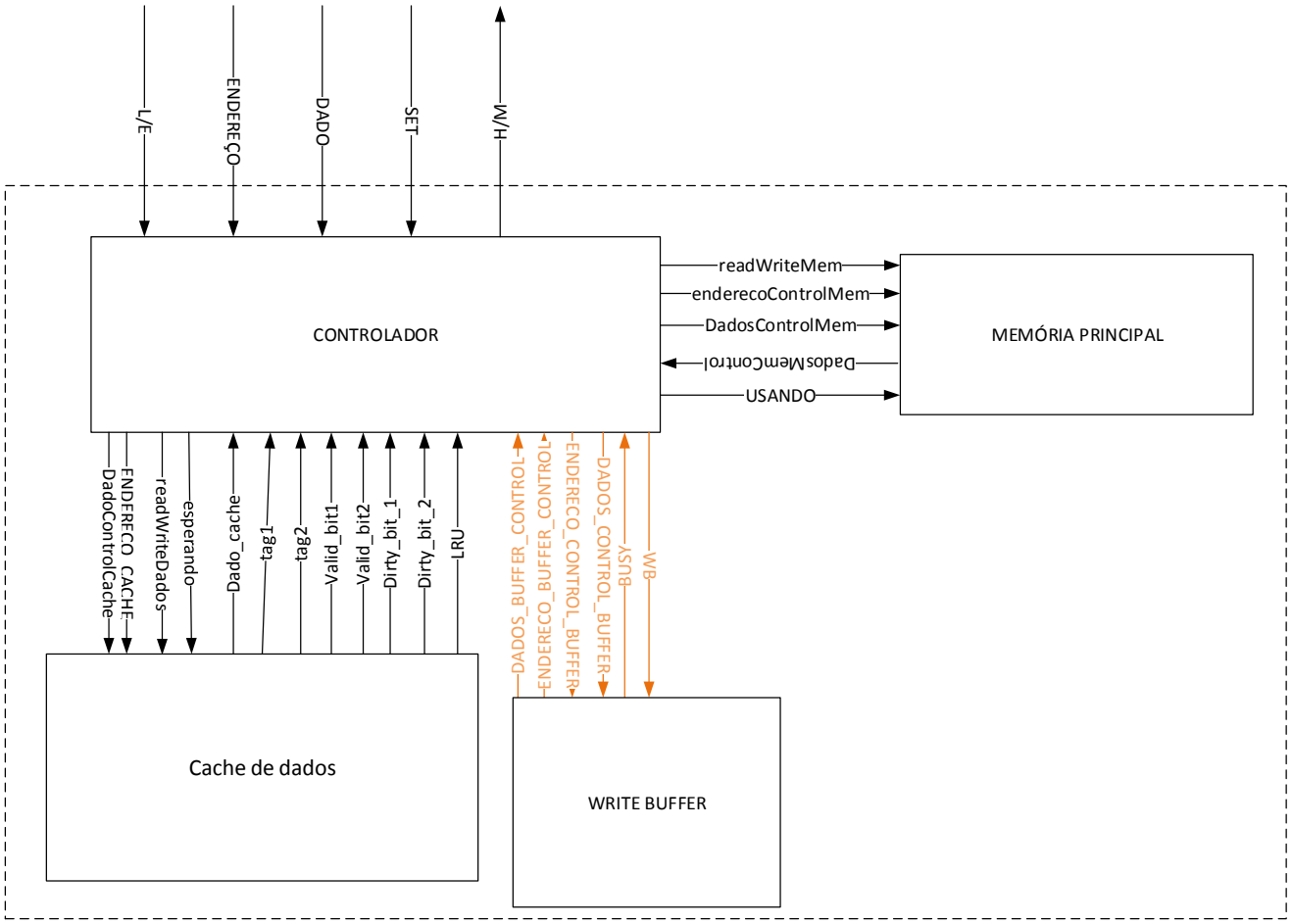


## 2.2. CACHE DE DADOS

O esquema geral do cache de dados segue o da figura 8. Também conta com um controlador, o cache propriamente dito e a memória principal. Porém, como adotou-se uma política de escrita de Write-Back, para tal optou-se por utilizar também um buffer (Write Buffer na figura).

Os sinais de entrada principais são o de L/E, que indica se deseja-se ler ou escrever um conteúdo no cache, o endereço de escrita ou leitura, o dado a ser escrito caso L/E = 0, o sinal set de clock, e o sinal de saída h/m que indica se houve hit ou miss no caso de se fazer uma leitura.

Figura 8: Diagrama de Blocos da hierarquia para o cache de dados




### 2.2.1. DESCRIÇÃO DOS MÓDULOS INTERIORES

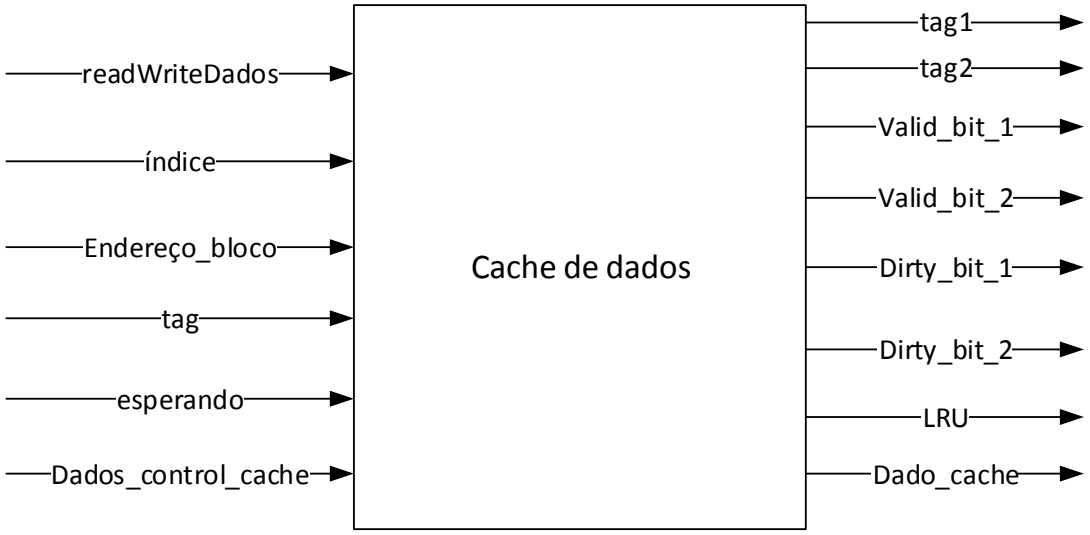
#### 2.2.1.1. Cache de Dados

O Cache de Dados tem como entrada os sinais de R/W, o índice, a tag, o endereço da palavra no bloco, a tag, o sinal de controle “esperando” e os dados que se deseja escrever na tag se for o caso. Como sinais de saída, há os valores das tags para os dois blocos de cada índice, os respectivos bits de validade, os respectivos bits sujos, o bloco usado mais anteriormente (LRU) e o dado lido da cache quando o caso.

Em termos de implementação, foi feito um array de 128 posições para cada bloco (bloco1 e bloco2), dois arrays de tag, bit de validade, bit sujo

referenciando o bloco 1 e o bloco 2 e um array de bits de 128 posições para os bits de LRU.

Figura 9:Cache de Dados



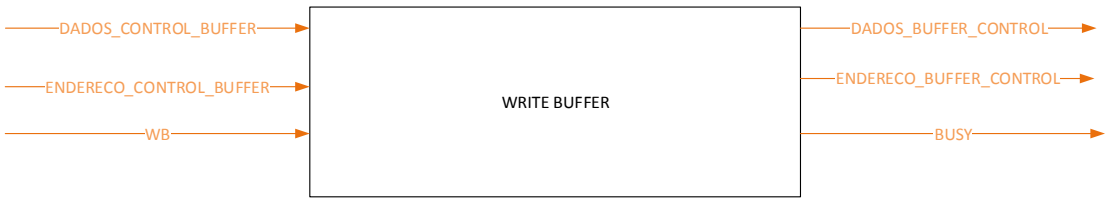

2.2.1.2. Write Buffer

O buffer de escrita tem como sinais de entrada os dados a serem escritos, o endereço em que o dado foi escrito e o sinal de controle WB usado pelo controlador para indicar quando que o buffer deve ser atualizado.

Os sinais de saída são os dados lidos do buffer, o endereço armazenado no buffer e em qual os dados devem ser armazenados, e um sinal “busy” que indica se ele está usado ou não (este é usado pelo controlador para saber quando pode escrever no buffer ou quando deve espera-lo mandar os dados para a memória).

Em termos de implementação, o buffer consiste em um vetor de uma palavra de 32 bits, um vetor de endereço de 16 bits

Figura 10: Write Buffer

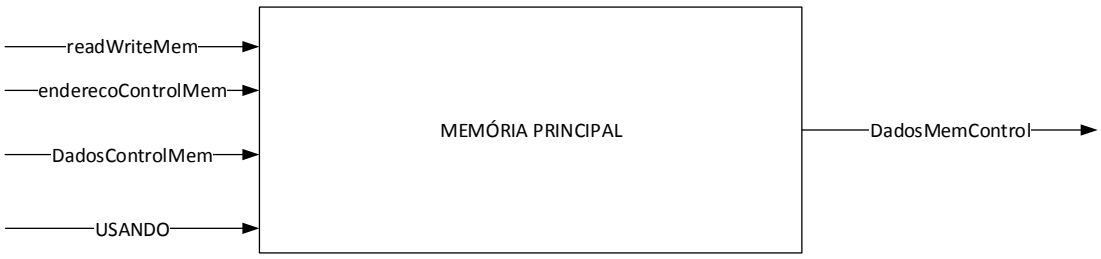



2.2.1.3. Memória Principal

A memória principal consiste em um array de 14 bits de posição, onde cada posição armazena uma palavra de 32 bits.

Os sinais de entrada são R/W, o endereço a ser acessado, os dados a serem armazenados quando o caso e o sinal de controle “usando”. Como saída é o sinal de dados lidos, quando o caso.

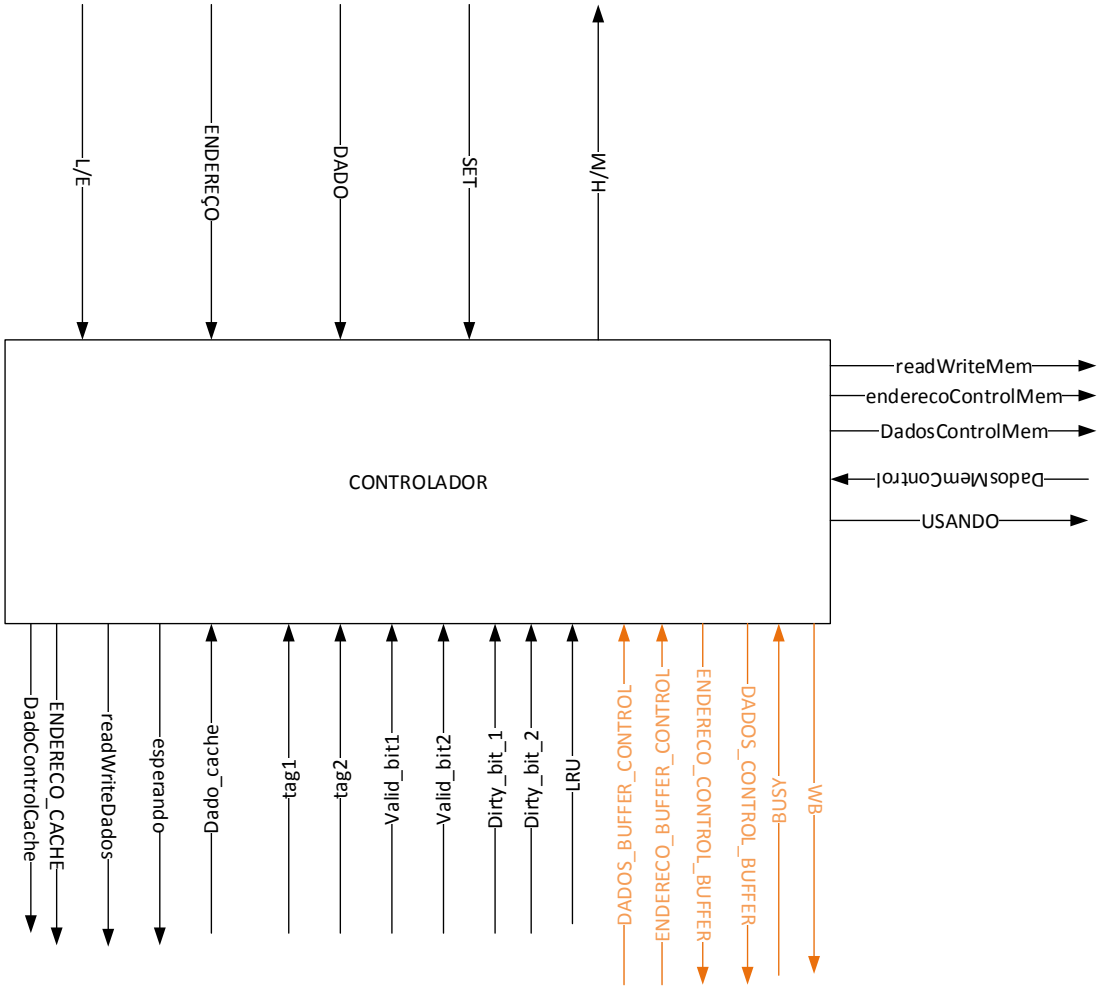
Figura 11: Memória Principal

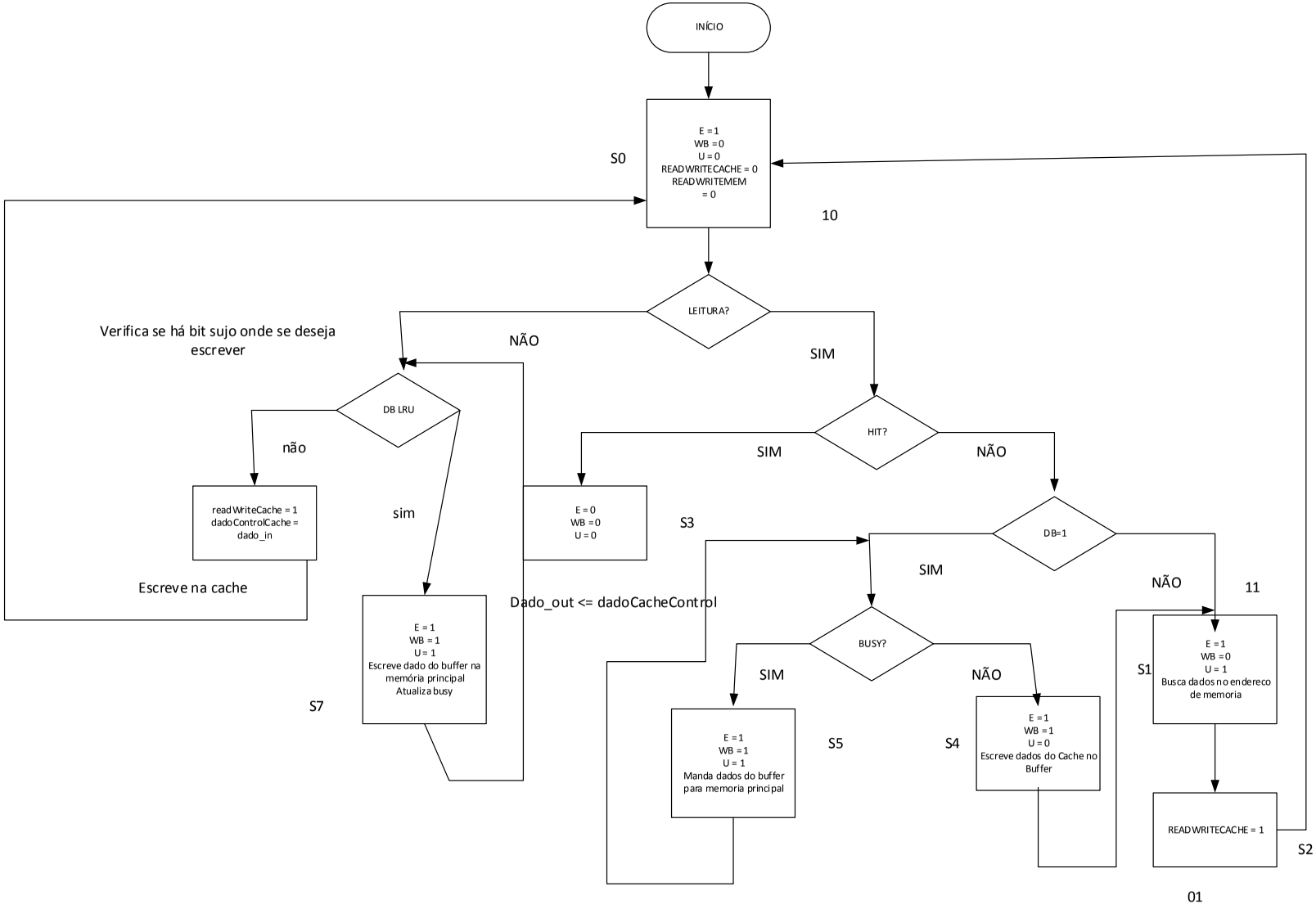



2.2.1.4. Controlador

O controlador projetado seguiu a ASM da figura 13 e o diagrama de blocos da figura 12.

Figura 12: Diagrama de Blocos




INÍCIO
E = 1
WB = 0
U = 0
READWRITECACHE = 0
READWRITEMEM = 0
S0
10
LEITURA?
NÃO
SIM
Verifica se há bit sujo onde se deseja escrever
DB LRU
não
sim
readWriteCache = 1
dadoControlCache = dado_in
Escreve na cache
E = 1
WB = 1
U = 1
Escreve dado do buffer na memória principal
Atualiza busy
S7
HIT?
SIM
NÃO
E = 0
WB = 0
U = 0
S3
Dado_out <= dadoCacheControl
DB=1
SIM
NÃO
11
BUSY?
SIM
NÃO
E = 1
WB = 1
U = 1
Manda dados do buffer para memoria principal
S5
E = 1
WB = 1
U = 0
Escreve dados do Cache no Buffer
S4
E = 1
WB = 0
U = 1
Busca dados no endereco de memoria
S1
READWRITECACHE = 1
S2
01

O controlador começa no estado default em que esperando = 1, usando = 0 e WB = 0. Caso seja exigida uma leitura, o controlador verifica se há um hit. Havendo um hit, é feita a leitura da cache e retorna-se ao estado default. Havendo um miss, são feitas duas verificações: i) avalia-se o dirty bit correspondente ao LRU, caso seja 0, busca-se os dados na memória principal e tem-se um comportamento análogo ao do cache de instruções, caso o dirty bit seja 1 é necessário fazer a segunda avaliação(ii); ii) avalia-se a situação do write buffer, caso ele esteja ocupado, suas informações são enviadas para a memória principal, caso esteja livre, são escritos nele as informações do cache. Passadas pelas duas avaliações, o conteúdo do cache é atualizado e lido.

Caso seja necessário uma escrita no cache de dados, avalia-se a posição de escrita: caso o dirty bit seja 0, escreve-se os dados no bloco do LRU. Caso o dirty bit seja 1, avalia-se a situação do Write Buffer: caso ele esteja ocupado, envia-se suas informações para a Memória Principal e recebe os dados enviados da cache. Caso não esteja ocupado, ele simplesmente recebe os dados da cache.

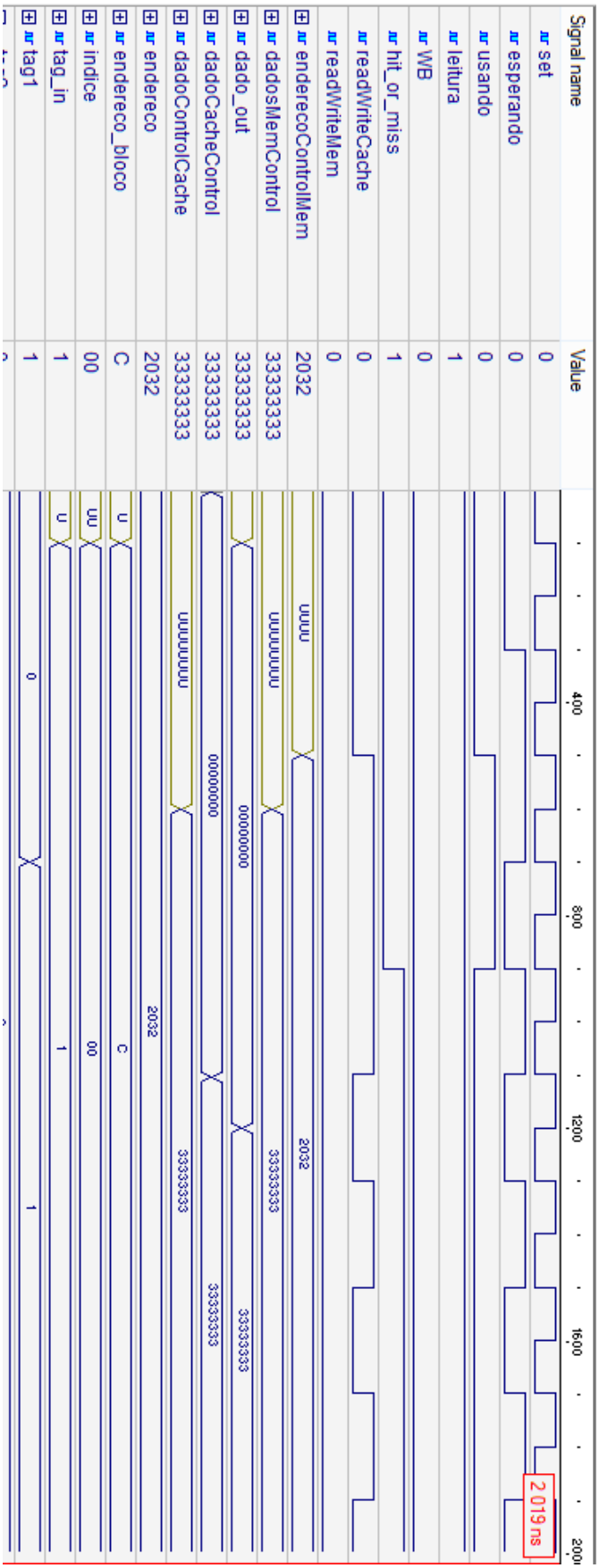
Sempre que se muda o conteúdo do Write buffer, é enviado para ele os dados que devem ser gravados, bem como o endereço que ele deve ser grava (tag+índice+bit_bloco+bits_offset).

O cache de dados interpreta os endereços recebidos da seguinte forma:

- Bits 0-1: bits de offset;
- Bits 2-5: bits de localização de palavra no bloco;
- Bits 6-12: bits de índice (128 posições);
- Bits 13-15: bits de tag.

2.2.2.SIMULAÇÃO

2.2.2.1. Leitura sem precisar do Write Buffer



3. BIBLIOGRAFIA

[1] Livro Texto da Disciplina.