Prática V - Parte I

LAOC II

Fevereiro, 2022

Autores:
Alex Meireles Santos Almeida
Vitor Theodoro Rocha Domingues

Centro Federal de Educação Tecnológica de Minas Gerais

Objetivo

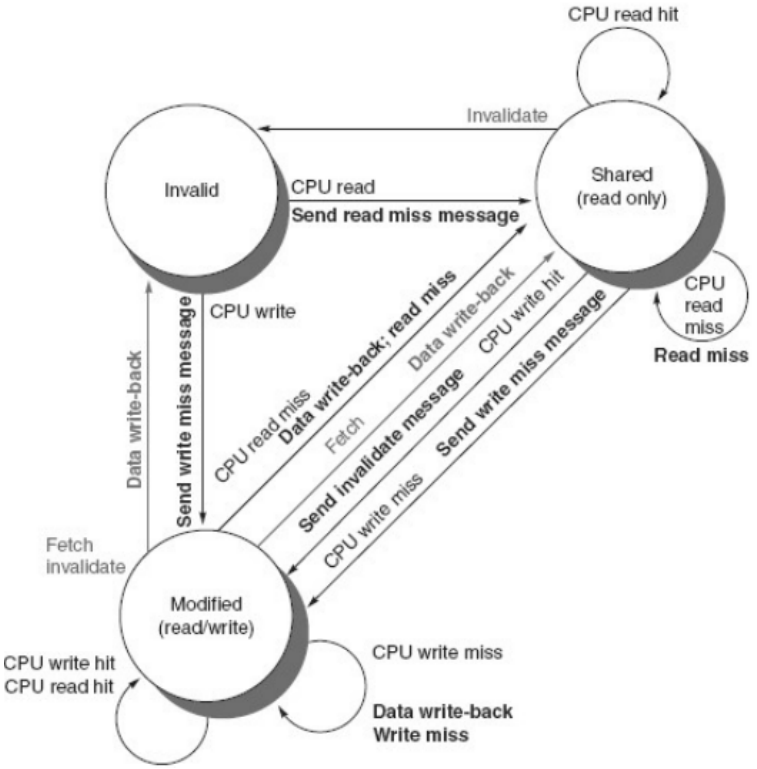
Esta prática tem a finalidade de exercitar os conceitos relacionados às máquinas de estados do protocolo Diretório.

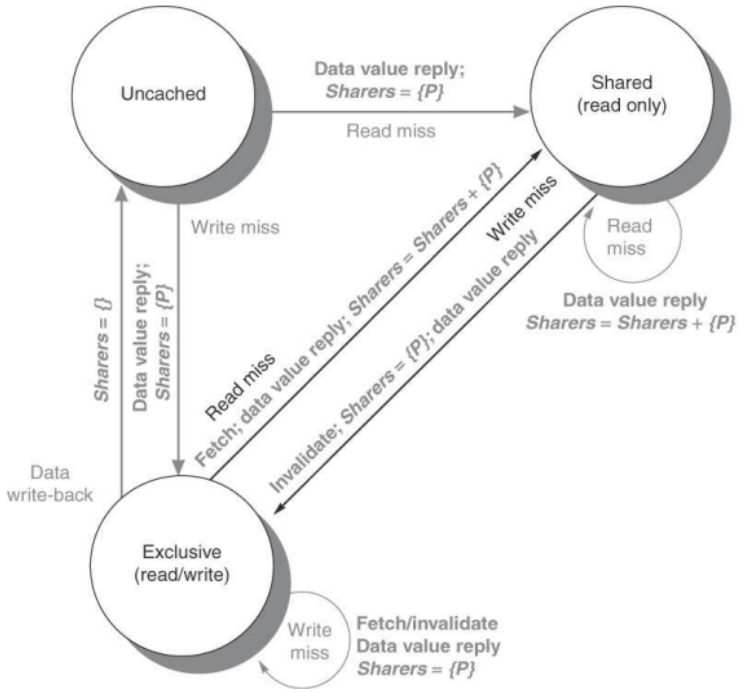
Introdução

A parte I da prática 4 consiste na implementação das máquinas de estado do protocolo Diretório, com o intuito de simular todas as transições das máquinas de estado e emitir com todas as suas mensagens e ações. A realização dos testes foram baseadas, em teoria, que a cache e a lista teriam o mesmo endereço.

Desenvolvimento

Para a implementação das máquinas de estados do protocolo Diretório, foi utilizado o diagrama abaixo como base:


CPU read hit
Invalidate
Invalid
Shared
(read only)
CPU read
Send read miss message
CPU write
Send write miss message
Data write-back
CPU read miss
Data write-back; read miss
Fetch
Data write-back
Send invalidate message
CPU write hit
Send write miss message
CPU write miss
CPU
read
miss
Read miss
Fetch
invalidate
Modified
(read/write)
CPU write hit
CPU read hit
CPU write miss
Data write-back
Write miss


Uncached
Data value reply;
Sharers = {P}
Read miss
Shared
(read only)
Write miss
Sharers = {}
Data value reply;
Sharers = {P}
Read miss
Fetch; data value reply; Sharers = Sharers + {P}
Write miss
Invalidate; Sharers = {P}; data value reply
Read
miss
Data value reply
Sharers = Sharers + {P}
Data
write-back
Exclusive
(read/write)
Write
miss
Fetch/invalidate
Data value reply
Sharers = {P}

O código consiste em três módulos: o TestBench, o Diretório e as máquinas de estado. Para cada módulo será mostrado abaixo as suas funcionalidades. Sendo assim:

- ## Diretório - Diretorio:

O módulo Diretorio, tem como objetivo chamar os TestBench e as máquinas de estado e é módulo principal para a realização da simulação a partir do Clock que é passado como entrada.

- ## Máquinas de estados - maquinaDeEstado:

Para a realização das transações, foi utilizado as seguintes variáveis:

ESTADOS	
INVALID	3’b001
SHARED	3’b010
MODIFIED	3’b011

SIGNAL	
empty	3’b000
ReadMiss	3’b001
ReadHit	3’b010
WriteMiss	3’b011
WriteHit	3’b100
Invalidate	3’b101

WRITEBACK	
Miss	2’b00
Hit	2’b01

Segue abaixo, dois exemplos de transições da máquina 1 e da máquina 2:

Esse caso acontece quando o testbench emite um sinal de Read miss e quando o estado da cache é inválida. Se entrar na condição, acontecerá uma transição do estado inválido para shared e emitirá um sinal de Read miss para a segunda máquina de estado que realizará a transição da lista(diretório).

```
if(stateCache == 3'b001 && WriteRead == 2'b00 && HitMiss == 2'b00) //INVALID - CPU Read
   begin
      newStateCache = 3'b010;             // Shared
      signal = 3'b001;                    // Read miss
   end
```

Com o sinal emitido na primeira máquina vai ter uma verificação se o estado que está na lista é inválido e se o sinal recebido é realmente Read Miss. Se entrar na condição, o estado será mudado para Shared.

```
if(stateDiretorio == 3'b001 && signal == 3'b001) //INVALID - ReadMiss
   begin

      newStateDiretorio = 3'b010; // SHARED

   end
```

- **Testbench - testbench:**

O módulo do testbench, tem como objetivo receber o código teste e passar esse código para a máquina de estado para verificação das transições. É passado para máquina, o estado da cache, o estado do diretório, se é read ou write e se deu hit ou miss.

O teste realizado é baseado apenas na máquina de estado, logo, quando é passado o estado da cache e do diretório é considerado que ambos os estados tenham o mesmo endereço. Isso foi estabelecido pois, para a transição ocorrer, primeiramente deve ser verificado se existe o mesmo endereço na cache e no diretório, para depois acontecer a transição na máquina de estado. Dessa forma, a verificação não é feita pela máquina de estado, por isso foi considerada em teoria que a cache e o diretório tenham o mesmo endereço para realização dos testes.

O código abaixo é um exemplo de como é feito o testbench.

```
// INVALID READ Miss
regHitMiss[0] = 2'b00;          // Miss
regWriteRead[0] = 2'b00;        // Read
regStateCache[0] = 3'b001;      // I
regStateDiretorio[0] = 3'b001;  // I

// INVALID WRITE Miss
regHitMiss[1] = 2'b00;          // Miss
regWriteRead[1] = 2'b01;        // Write
regStateCache[1] = 3'b001;      // I
regStateDiretorio[1] = 3'b001;  // I
```

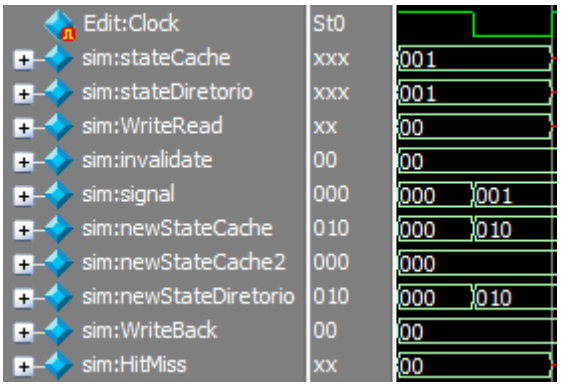
- **Simulação:**

1. Invalid -> Read Miss

Estado da cache: Invalid
Estado do diretório: Invalid
Sinal: Read Miss

Novo estado da cache: Shared
Novo estado do diretório: Shared
WriteBack: Miss

```
// INVALID READ Miss
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[0] = 2'b00;         // Read
regStateCache[0] = 3'b001;       // I
regStateDiretorio[0] = 3'b001;   // I
```

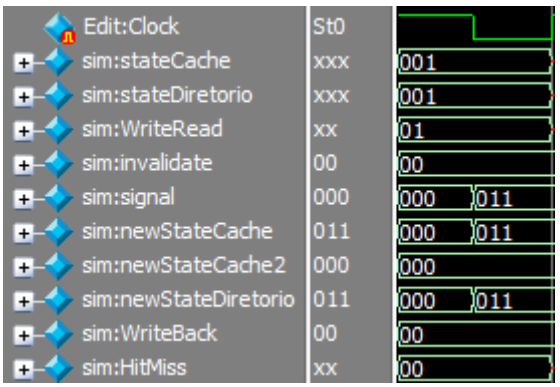


2. Invalid -> Write Miss

Estado da cache: Invalid
Estado do diretório: Invalild
Sinal: Write Miss

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Miss

```
// INVALID WRITE Miss
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[0] = 2'b01;         // Write
regStateCache[0] = 3'b001;       // I
regStateDiretorio[0] = 3'b001;   // I
```

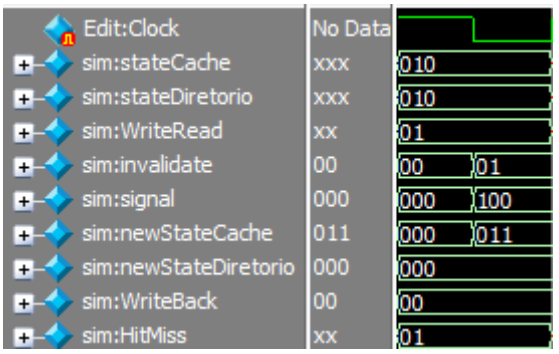


3. Shared -> Write Hit

Estado da cache: Shared
Estado do diretório: Shared
Sinal: Write Hit

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Miss

```
// SHARED WRITE Hit
regHitMiss[0] = 2'b01;            // Hit
regWriteRead[0] = 2'b01;          // Write
regStateCache[0] = 3'b010;        // S
regStateDiretorio[0] = 3'b010;    // S
```

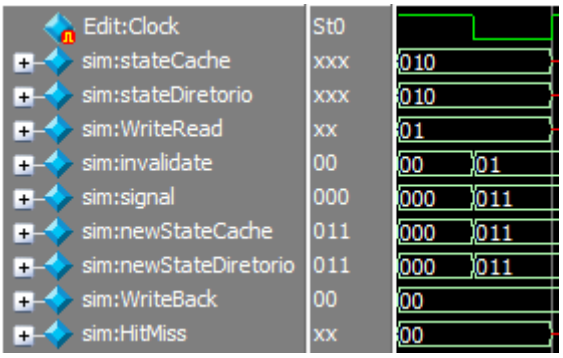


4. Shared -> Write Miss

Estado da cache: Shared
Estado do diretório: Shared
Sinal: Write Miss

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Miss

```
// SHARED WRITE Miss
regHitMiss[0] = 2'b00;            // Miss
regWriteRead[0] = 2'b01;          // Write
regStateCache[0] = 3'b010;        // S
regStateDiretorio[0] = 3'b010;    // S
```

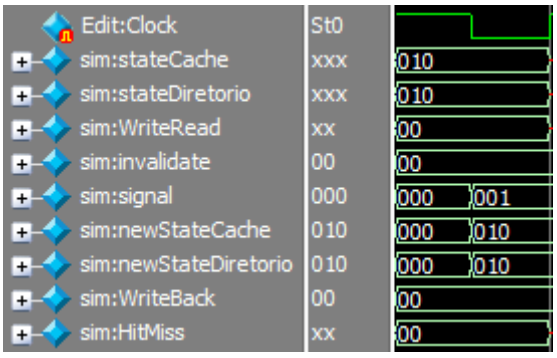


5. Shared -> Read Miss

Estado da cache: Shared
Estado do diretório: Shared
Sinal: Read Miss

Novo estado da cache: Shared
Novo estado do diretório: Shared
WriteBack: Miss

```
// SHARED READ Miss
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[] = 2'b00;          // Read
regStateCache[0] = 3'b010;       // S
regStateDiretorio[0] = 3'b010;   // S
```

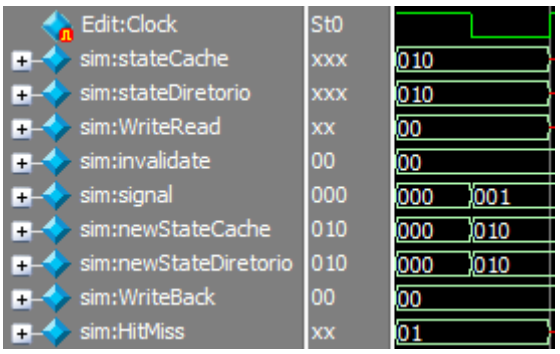


6. Shared -> Read Hit

Estado da cache: Shared
Estado do diretório: Shared
Sinal: Read Hit

Novo estado da cache: Shared
Novo estado do diretório: Shared
WriteBack: Miss

```
// SHARED READ Hit
regHitMiss[0] = 2'b01;           // Hit
regWriteRead[0] = 2'b00;         // Read
regStateCache[0] = 3'b010;       // S
regStateDiretorio[0] = 3'b010;   // S
```

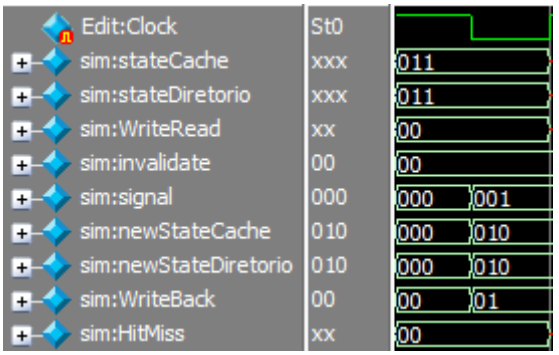


7. Modified -> Read Miss

Estado da cache: Modified
Estado do diretório: Modified
Sinal: Read Miss

Novo estado da cache: Shared
Novo estado do diretório: Shared
WriteBack: Hit

```
// MODIFIED READ Miss
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[0] = 2'b00;         // Read
regStateCache[0] = 3'b011;       // M
regStateDiretorio[0] = 3'b011;   // M
```

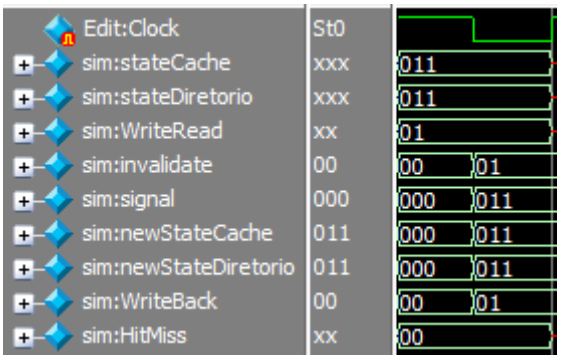


8. Modified -> Write Miss

Estado da cache: Modified
Estado do diretório: Modified
Sinal: Write Miss

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Hit

```
// MODIFIED WRITE Miss
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[0] = 2'b01;         // Write
regStateCache[0] = 3'b011;       // M
regStateDiretorio[0] = 3'b011;   // M
```

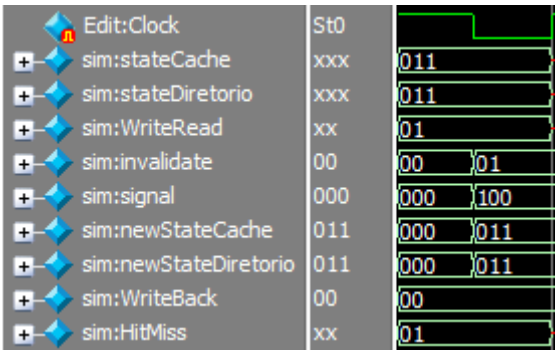


9. Modified -> Write Hit

Estado da cache: Modified
Estado do diretório: Modified
Sinal: Write Hit

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Miss

```
// MODIFIED WRITE Hit
regHitMiss[0] = 2'b01;            // Hit
regWriteRead[0] = 2'b01;          // Write
regStateCache[0] = 3'b011;        // M
regStateDiretorio[0] = 3'b011;    // M
```

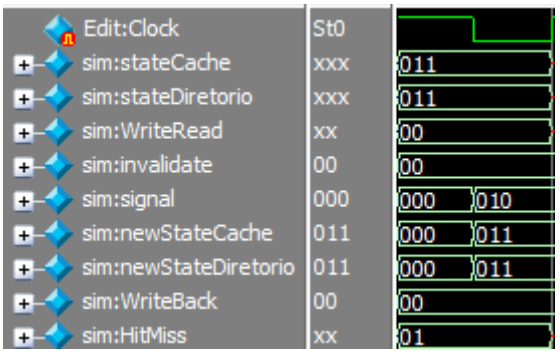


10. Modified -> Read Hit

Estado da cache: Modified
Estado do diretório: Modified
Sinal: Read Hit

Novo estado da cache: Modified
Novo estado do diretório: Modified
WriteBack: Miss

```
// MODIFIED READ Hit
regHitMiss[0] = 2'b01;            // Hit
regWriteRead[0] = 2'b00;          // Read
regStateCache[0] = 3'b011;        // M
regStateDiretorio[0] = 3'b011;    // M
```



. Casos Especiais:

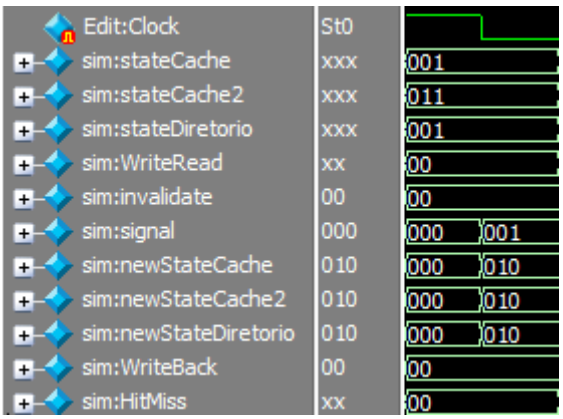
Para mostrar as transições desses casos foi necessário a criação de uma variável cache2. Essa variável é apenas um símbolo para o funcionamento desses casos, não foi criado nenhuma cache que está relacionada ao protocolo, apenas uma variável para ajudar a mostrar essa transição.

1. INVALID -> Read Miss

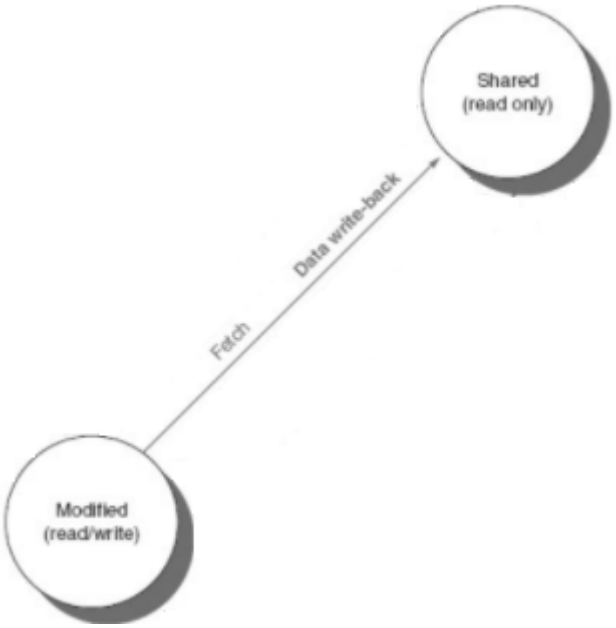
Estado da cache: Invalid
Estado da cache2: Modified
Estado do diretório: Invalid
Sinal: Read Miss

Novo estado da cache: Shared
Novo estado da cache2: Shared
Novo estado do diretório: Shared
WriteBack: Miss

```
INVALID READ MISS - CACHE2 = MODIFIED -
regHitMiss[0] = 2'b00;           // Miss
regWriteRead[0] = 2'b00;         // Read
regStateCache[0] = 3'b001;       // I
regStateDiretorio[0] = 3'b001;   // I
regStateCache2[0] = 3'b011;      // M
```



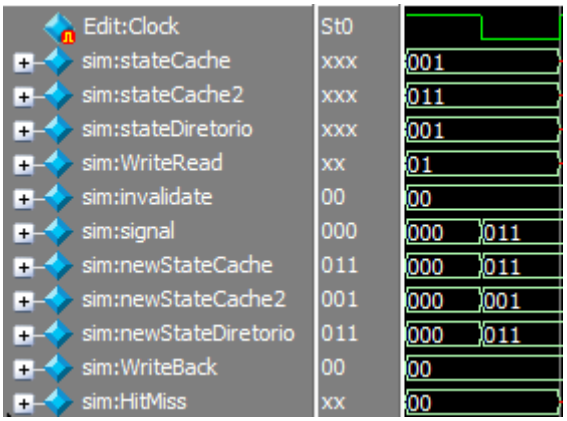
2. INVALID -> Write miss



Estado da cache: Invalid
Estado da cache2: Modified
Estado do diretório: Invalid
Sinal: Write Miss

Novo estado da cache: Modified
Novo estado da cache2: Invalid
Novo estado do diretório: Modified
WriteBack: Miss

```
INVALID WRITE MISS - CACHE 2 = MODIFIED -
regHitMiss[0] = 2'b00;            // Miss
regWriteRead[0] = 2'b01;          // Write
regStateCache[0] = 3'b001;        // I
regStateDiretorio[0] = 3'b001;    // I
regStateCache2[0] = 3'b011;       // M
```

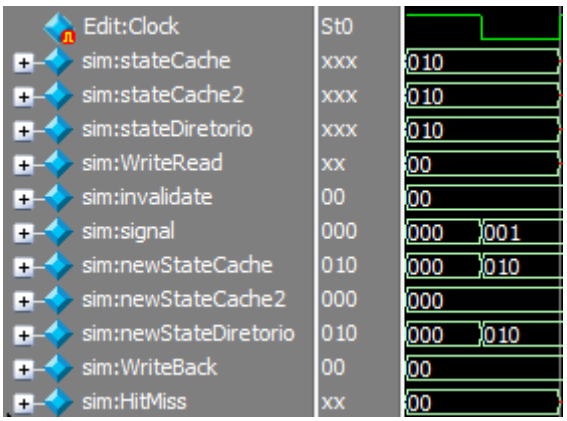


3. Modified -> Read Hit

Estado da cache: Shared
Estado da cache2: Shared
Estado do diretório: Shared
Sinal: Read Hit

Novo estado da cache: Modified
Novo estado da cache2: Invalid
Novo estado do diretório: Modified
WriteBack: Miss

```
SHARED WRITE HIT - CACHE2 = SHARED
regHitMiss[0] = 2'b00;              // Miss
regWriteRead[0] = 2'b00;            // Read
regStateCache[0] = 3'b010;          // S
regStateDiretorio[0] = 3'b010;      // S
regStateCache2[0] = 3'b010;         // S
```



- ## Dificuldades:

A criação apenas da máquina de estado sem utilização de uma cache e o endereço do bloco foi uma grande dificuldade por conta de casos específicos que necessitavam de estados de outras cache para realizar a transição. E a falta de um caso teste, fez que não tivesse um direcionamento certo na hora de criar o código.

- ## Sugestões:

Disponibilizar o código teste ajudaria bastante para o direcionamento na criação do código.