ARQUITETURA DE COMPUTADORES

Prática 6

Pretende-se analisar a hierarquia de memória do processador ilustrado na Figura 1. O processador funciona a uma frequência de 2GHz e é implementado por uma arquitetura *pipelined* que suporta o subconjunto do conjunto de instruções RV64G para operações com inteiros e virgula flutuante, utilizando **32 bits para instruções** e **64 bits para dados**. A unidade de armazenamento do processador é composta por um banco de 32 registos de inteiros de 64 bits.

A hierarquia de memória é composta por uma cache de instruções e uma cache de dados, ambas ligadas a uma memória principal partilhada. A configuração de todos os componentes da hierarquia de memória é apresentada na Tabela 1.

Resolva os exercícios abaixo de acordo com informação fornecida no enunciado.

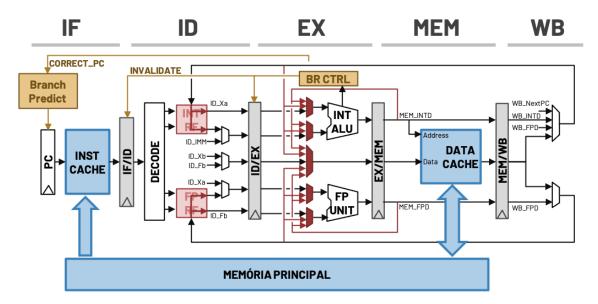


Figura 1 - Arquitetura do Processador

Tabela 1 – Configuração da Hierarquia de Memória.

CACHE INSTRUÇÕES	CACHE DADOS	MEMÓRIA PRINCIPAL
Mapeamento Direto Capacidade (dados) 64B Linhas de dados 16B	2 Vias de Associatividade Capacidade (dados) 128B Linhas de dados 16B Política de substituição: - First-In First-Out (FIFO) Política de escrita: - Write Back, Write Allocate	Capacidade 256MB
Latência T₁=0.9ns	Latência T _D =1.2ns	Latência T _M = 20ns

Exercício 1

De acordo com as configurações apresentadas na Tabela 1, para cada uma das caches indique a decomposição da palavra de endereço (*Tag, Index e Offset*) e determine o número de ciclos de relógio necessários para realizar o acesso às caches.

Exercício 2

Considere o código *Assembly* apresentado abaixo. Simule a execução do código na arquitetura da Figura 1 e apresente: **(1)** o preenchimento das caches de instruções e dados após o final da execução; e **(2)** o número total de *Hits* e *Misses* em cada cache.

NOTA: Para representar valores de dados utilize a representação da indexação ao vetor correspondente (ou seja, *A[0]*, *A[1]*, ... e *B[0]*, *B[1]*,).

```
000h|
         ori
                  x5, x0, N
                                 ; N=16
                                 ; N-1 (15)
004h|
         addi
                  x5, x5, -1
                                 ; 64-bit data
008h|
         sll
                  x6, x5, 3
                  x10, x6, 80h
00Ch|
         addi
                                 ; vector A
010hl
         addi
                  x11, x6, 100h ; vector B
014h|
         addi
                  x12, x0, 200h ; result
018h|
                                 ; f12=0.0
         fcvt.d.w f12, x0
01Ch| L: jal
                  x1, R
020h|
         addi
                  x10, x10, -8
                  x11, x11, -8
024h|
         addi
028h|
                  x5, x5, -1
         addi
02Ch|
         bge
                  x5, x0, L
         fsd
                  f12, 0(x12)
030hl
                                                           CACHE INSTRUÇÕES
424h| R: fld
                  f10, 0(x10)
                                                           TAG
                                                                       DATA
428h|
         fld
                  f11, 0(x11)
                                                  00
                  f11, f11, f10
42Chl
         fmul.d
                  f12, f12, f11
                                                  01
430h|
         fadd.d
434h|
         jalr
                  x0, x1, 0
                                                  10
                                                  11
```

	CACHE DADOS							
	V	TAG	DATA		V	TAG	DATA	
00								
01								
10								
11								

Exercício 3

Calcule a taxa de sucesso no acesso (*Hit Rate*) de cada cache (HR_I e HR_D) com base nos resultados obtidos no Exercício 2. Calcule os tempos médios de acesso (em nanossegundos) às instruções (T_I) e aos dados (T_D) da hierarquia de memória.

Exercício 4

Considere um processador que opera a uma frequência f_{clk} = 1 GHz. Este processador contêm duas caches de dados (L1 e L2). A configuração da hierarquia de memória é apresentada na Tabela 1.

Considere que os barramentos que ligam o processador, caches e a memória principal são de 32-bits para os endereços e de 64-bits para os dados.

L1 CACHE DADOS	L2 CACHE DADOS	MEMÓRIA PRINCIPAL
Mapeamento Direto	2 Vias de Associatividade	Capacidade 256MB
Capacidade (dados) 64B	Capacidade (dados) 128B	
Linhas de dados 16B	Linhas de dados 16B	
Política de substituição:		

Política de escrita: Write Back, Write Allocate

Latência T_{L2}= 5T_{clk}

Tabela 2 – Configuração da Hierarquia de Memória.

Durante a operação, o processador executa o seguinte troço de código utilizando operandos em vírgula flutuante de 64-bits (double):

O compilador não realizou qualquer optimização, pelo que os operandos são lidos na mesma ordem que aparecem no programa (exemplo: para uma instrução de 2 operandos Dest = Op1 OP Op2, a ordem de acesso será: Op1, Op2, Dest). As variáveis são alocadas em regiões adjacentes e por

```
#define REPEATS 10
#define NUM_ELEMENTS ??

double A[16];
register int i, r;

...
for (r=0; r<REPEATS; r++)
   for (i=0; i<=NUM_ELEMENTS; i+=2)
        A[i+2] = A[i] + A[i+1];</pre>
```

Latência T_{L1}=T_{clk}

ordem crescente de endereços (início em 00..00h).

Latência T_M= 50T_{clk}

- a) Descreva a organização das caches de dados (L1 e L2), identificando a forma como os bits de endereço são usados para aceder a cada posição.
- b) Decomponha a palavra de endereço deA[0] a A[15] nos campos correspondentes à

tag, index e offset da cache L1 e da cache L2.

- c) Determine a taxa de sucesso (*hit-rate*) das caches L1 e L2 para este troço de programa considerando os seguintes valores de NUM ELEMENTS:
 - 1. NUM ELEMENTS=4
 - 2. NUM ELEMENTS=12
- d) Calcule o tempo médio de acesso à memoria (com e sem L2 cache) para os casos indicados na linha anterior. Calcule o speedup para estes casos e comente o resultado obtido.