Arquitectura de Computadores II
3ª Frequência e Exame

Departamento de Informática
Universidade de Évora

5 de Janeiro de 2023

- Os símbolos à esquerda de cada pergunta identificam a prova ou provas a que ela pertence:
  ♣ assinala as perguntas do exame; ♢ assinala as perguntas da frequência.
- Nas perguntas comuns ao exame e à 3ª frequência, a primeira cotação indicada diz respeito ao exame, e a segunda diz respeito à frequência.
- Indique todos os cálculos efectuados e todas as fórmulas usadas

Perguntas rápidas

♣ **1.** [0,5 valores] Sabendo que a execução de um programa no processador X necessita do dobro do tempo da execução do mesmo programa no processador Y, pode concluir que o desempenho de X para esse programa é inferior ao de Y?

♣ **2.** [0,5 valores] Quando, no *pipeline* RISC-V, uma instrução é atrasada um ciclo devido a um conflito de dados que a envolve, é a entrada da instrução no *pipeline* que é atrasada, ou a instrução fica um ciclo de relógio adicional nalgum andar do *pipeline*?

♣ ♢ **3.** [0,5/1 valores] Numa cache 8-*way set associative* com 64 conjuntos, em quantas posições pode ser colocado um dado bloco?

♢ **4.** [1 valor] Em geral, quando é necessário colocar uma nova página virtual em memória física, qual é a estratégia usada para escolher a página física cujo conteúdo irá ser substituído?

♢ **5.** [1 valor] Onde se pode encontrar informação sobre qual a página física em que se encontra uma dada página virtual de um processo?

♣ ♢ **6.** [0,5/1 valores] Tendo em conta a lei de Amdahl, quando um programa é paralelizado e executado em k processadores, o *speedup* que se pode esperar obter, em relação à sua execução sequencial, é inferior a k, igual a k, ou superior a k?

Desempenho

♣ **7.** [3 valores] Um processador, cujo relógio tem uma frequência de 2 GHz, executa os 100 milhões de instruções de um programa em 0,15 s. Dada a distribuição das instruções do programa, apresentada abaixo, calcule o CPI global correspondente e o CPI das instruções da classe B.

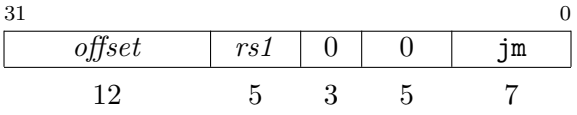
Classe	A	B	C	D
%	25	30	30	15
CPI	3	?	3	1

(CONTINUA. . .)

Implementação RISC-V monociclo

♣ **8.** [4 valores] Pretende-se que a implementação RISC-V monociclo da Figura 1 suporte a execução da instrução `jm` (*jump memory*), que é uma instrução tipo-I com dois argumentos:

`jm offset(rs1)`



Esta é uma instrução de salto incondicional. O endereço da instrução executada a seguir a esta instrução encontra-se na memória, no endereço obtido somando `offset` ao conteúdo do registo `rs1`.

(a) Quais das unidades funcionais existentes (incluindo *multiplexers*) serão usadas na execução desta instrução? (Identifique os *multiplexers* com a ajuda dos seus sinais de controlo, e.g., MemtoReg.)

(b) Que unidades funcionais (incluindo *multiplexers*) e que sinais de controlo é necessário acrescentar?

Se não for necessário fazer qualquer alteração à implementação, explique brevemente qual será funcionamento do processador na execução da instrução.

(As alterações à implementação que considerar necessário fazer podem ser só apresentadas na Figura 1.)

(c) Na execução desta instrução, quais os valores que os vários sinais de controlo deverão ter e qual a operação realizada pela ALU? (Não é necessário apresentar o valor de ALUOp.)

Pipeline RISC-V de 5 andares

Use como referência o *pipeline* da Figura 2. Tenha, no entanto, em atenção, a caracterização adicional do funcionamento do *pipeline* feita na pergunta.

♣ **9.** [2,5 valores] Identifique todas as dependências (de dados) existentes no código à direita, indicando os registos envolvidos, e diga quais as que constituem conflitos.

(Assuma que `addi` e `slti` calculam os seus resultados no andar EX e que, em termos do que acontece no *pipeline*, `bne` funciona de modo semelhante a `beq`.)

```
1.  volta: addi t0, t0, 16
2.         lw   t1, 4(t0)
3.         slti t2, t1, 0
4.         beq  t2, zero, salta
5.         sub  t1, zero, t1
6.         sw   t1, 4(t0)
7.  salta: bne  t1, zero, volta
```

Cache

♣ ♢ **10.** Considere um sistema com palavras e endereços de 32 bits, blocos com 64 *bytes*, e com uma cache 8-*way set associative* com 1024 conjuntos e estratégia *copy back*.

Durante a execução de um programa, neste sistema, é lido o valor que está na posição de memória com endereço 192 009.

(a) [1,5/2,5 valores] Indique a palavra e o bloco a que pertence o *byte* que tem aquele endereço, o índice da posição da cache em que o bloco poderá estar, e o *tag* correspondente.

(b) [1,5/2,5 valores] Mostre o conteúdo da posição da cache onde o bloco ficará, tão completo quanto possível, assumindo que este é o primeiro acesso feito pelo programa a este bloco.

(CONTINUA. . .)

Memória virtual

♣ ◇ **11.** [2/3 valores] A tabela de páginas cujo conteúdo é parcialmente mostrado pertence a um processo que corre num processador com um TLB *direct mapped*, com capacidade para 4 traduções, e estratégia *write-through*. Este processo executa a sequência de operações seguinte:

1. Leitura de uma posição de memória da página virtual 11;
2. Leitura de uma posição de memória da página virtual 9;
3. Escrita de uma posição de memória da página virtual 9;
4. Leitura de uma posição de memória da página virtual 5;
5. Escrita de uma posição de memória da página virtual 7.

Tabela de páginas

Índice	*Dirty*	Pág. física
	...	
4	0	11
5	0	73
6	1	39
7	0	22
8	0	9
9	0	55
10	0	91
11	1	17
	...	

- Assuma que o TLB inicialmente está vazio e, para cada operação, indique o índice da posição do TLB que será usada, o *tag*, se há um *hit* ou um *miss* e, quando aplicável, a página virtual cuja tradução será substituída.
- Apresente o conteúdo final do TLB, tão completo quanto possível.
- Se o conteúdo de alguma posição da tabela de páginas é alterado devido às operações executadas, apresente o novo conteúdo dessa ou dessas posições.

♣ ◇ **12.** [1,5/2 valores] Se um endereço virtual tiver 36 bits, uma página tiver 8 KB, e cada tabela de segundo nível de uma tabela de páginas com dois níveis tiver 4096 posições, quantas posições terá a tabela de primeiro nível?

Multiprocessamento

13. Num sistema multiprocessador de memória partilhada, pretende-se implementar a função `set`, com um argumento (diferente de 0), que atribui o valor do argumento à variável global `counter`, se e só se a variável tem o valor 0. O valor devolvido pela função é o valor que encontra em `counter`, o que permite, posteriormente, saber se a função atribuiu, ou não, o valor do argumento à variável.

O funcionamento pretendido da função corresponde às versões C e RISC-V, abaixo. A versão RISC-V recebe o argumento no registo `a0` e devolve o resultado também no registo `a0`.

```
int set(int value)
{
  int current = counter;

  if (current == 0)
    counter = value;

  return current;
}
```

```
1.   set: lw   t0, counter
2.        bne  t0, zero, end
3.        sw   a0, counter
4.   end: addi a0, t0, 0
5.        jalr zero, 0(ra)
```

◇ (a) [3 valores] Considere uma situação em que o valor de `counter` é 0 e a função é executada em dois processadores, num com argumento 10, e no outro com argumento 20.

Recorrendo à versão RISC-V da função, mostre que é possível ambas as chamadas devolverem o valor 0 (o que significa que ambas atribuíram o valor do seu argumento a `counter`).

♣ ◇ (b) [2/3 valores] Apresente uma versão RISC-V da função que garanta que, mesmo em caso de execuções simultâneas, só uma chamada da função atribui um valor a `counter` e que os valores devolvidos reflectem essa realidade.

Nome: _____ Número: _____

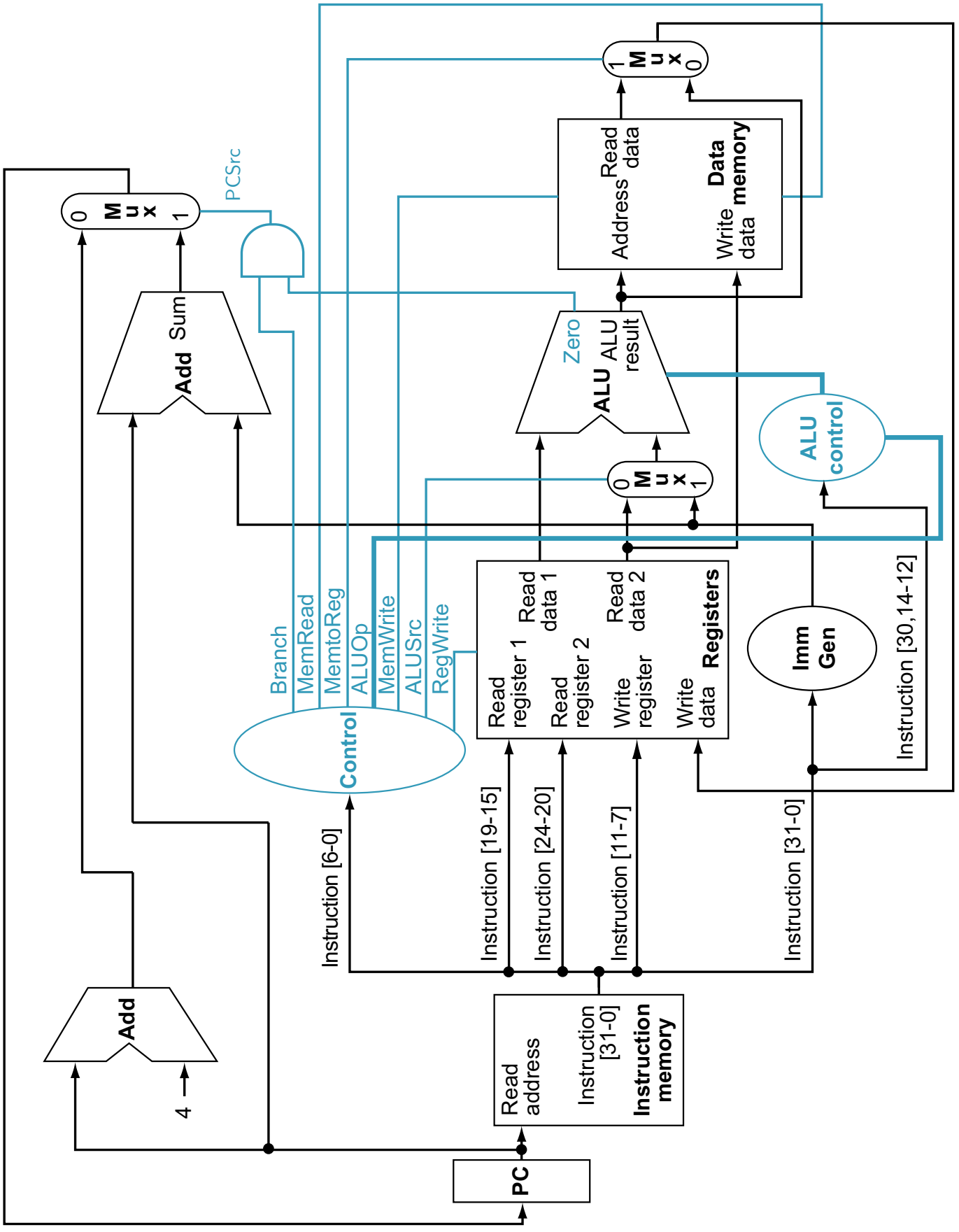


Figura 1: Diagrama de blocos da implementação RISC-V monociclo

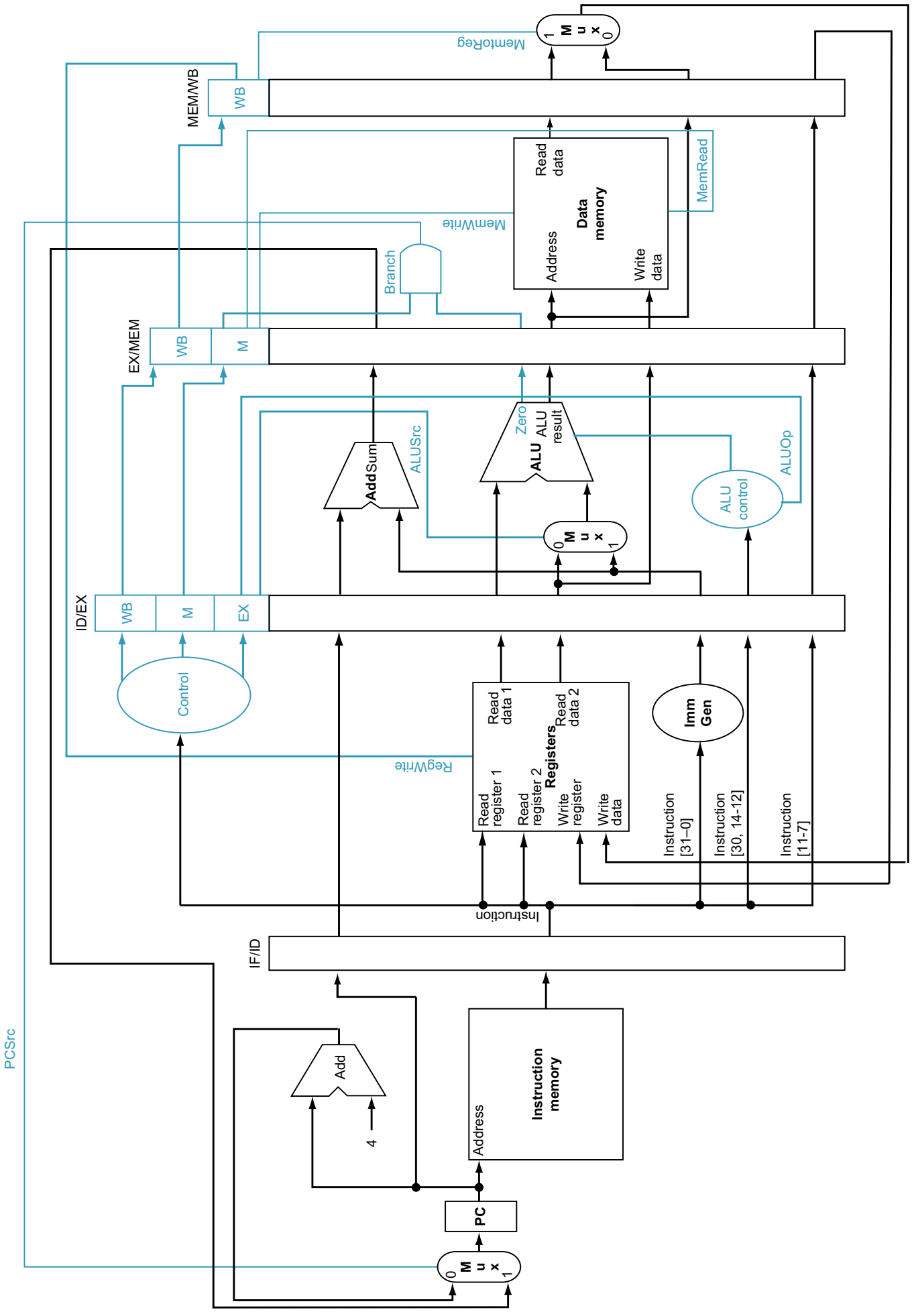


Figura 2: Diagrama de blocos do *pipeline* RISC-V