Arquitectura de Sistemas e Computadores II
1ª Frequência

Departamento de Informática ◇ Universidade de Évora

29 de Outubro de 2014

Indique todos os cálculos efectuados

Perguntas rápidas

1. [0,5 valores] Sabendo que a frequência do relógio do processador X é superior à do relógio do processador Y, pode concluir que o desempenho de X é superior ao de Y?

2. [0,5 valores] Seja m uma instrução com um *delay slot*. A instrução no *delay slot* de m é executada sempre que m é executada?

3. [0,5 valores] Se a execução de um programa é interrompida devido à ocorrência de uma excepção ou de uma interrupção, o que acontece ao programa quando o tratamento da excepção ou da interrupção termina?

4. [0,5 valores] Que tipo de conflito existiria se um *issue packet* do *pipeline* MIPS com *double issue* estático contivesse duas instruções de acesso à memória?

Desempenho

5. [3 valores] Um programa é executado no processador A, cujo relógio tem um período de 0,5 ns. Na execução do programa são executados 1000 milhões de instruções, divididas por três classes de acordo com a distribuição seguinte:

Classe	Aritméticas	Saltos condicionais	Acesso à memória
CPI	1	2	3
%	60	20	20

No processador B, que implementa a mesma arquitectura, foi possível baixar o período do relógio para 250 ps. No entanto, na execução do programa neste processador, verifica-se que o *tempo* que demoram as instruções de acesso à memória é *exactamente o mesmo* que demoram no processador A.

Qual o processador mais rápido para o programa e qual o *speedup* que ele oferece em relação ao outro?

Implementação MIPS monociclo

Para este grupo, use como referência a implementação monociclo da Figura 1.

6. Pretende-se incluir a instrução `ji` (*jump indirect*) na implementação MIPS monociclo. A instrução `ji` é uma instrução tipo-I com dois argumentos:

```
ji offset(rs)
```

	`ji`	*rs*	0	*offset*
bits	6	5	5	16

O efeito desta instrução é provocar o salto da execução para a instrução cujo endereço é obtido somando `offset`, que pode ser negativo, ao conteúdo de `rs`.

(a) [2,5 valores] Quais das unidades funcionais existentes serão usadas para a execução desta instrução e que unidades funcionais é necessário acrescentar?

(b) [2,5 valores] Que sinais de controlo é necessário acrescentar e quais os valores que os vários sinais de controlo deverão ter para a execução desta instrução?

(Não precisa de indicar o valor de ALUOp, basta dizer qual será a função da ALU na execução desta instrução.)

Se considerar necessário fazer alguma alteração ao caminho de dados, apresenta-a na Figura 1.

Pipeline MIPS de 5 andares

Para este grupo, use como referência o *pipeline* da Figura 2. Tenha, no entanto, em atenção as caracterizações do funcionamento do *pipeline* feitas nas várias alíneas.

7. [1 valor] Considere, agora, a inclusão da instrução `ji` na implementação *pipelined* do MIPS.
Em que andar estará a instrução no ciclo em que o endereço do destino do salto fica disponível? Quantos *delay slots* deveria ter esta instrução para que não fosse necessário introduzir atrasos nem apagar instruções do *pipeline*?

8. Pretende-se executar o código MIPS seguinte de modo a que o seu efeito seja exactamente o que teria se fosse executado na implementação monociclo do processador. No fim da execução do código, os valores nos registos usados não são importantes, excepto o do registo `$v0`.

```
1.           or    $v0, $0, $0
2.    ciclo: lw    $t0, 0($a1)
3.           sw    $t0, 0($a0)
4.           beq   $t0, $0, fim
5.           addi  $v0, $v0, 1
6.           addiu $a0, $a0, 4
7.           addiu $a1, $a1, 4
8.           beq   $0, $0, ciclo
9.    fim:   jr $ra
```

(a) [2 valores] Identifique todas as dependências (de dados) existentes no código apresentado.

(b) [2 valores] Simule a execução do código desde o início e até terminar a primeira execução da instrução 8. (Não precisa de considerar as instruções que entram no *pipeline* após o início da execução desta instrução.) Considere um processador com *forwarding*, com decisão dos saltos condicionais no andar EX, com previsão perfeita do resultado das instruções de salto condicional e sem *delay slots*. Apresente a evolução do estado do *pipeline* durante a execução, indicando todos os atrasos introduzidos e todos os pontos onde foi necessário o *forwarding* de algum valor, identificando claramente entre que andares o *forwarding* foi feito.

Quantos ciclos de relógio são necessários para executar o código nas condições acima?

Quantos ciclos de relógio seriam necessários para executar o código se fossem executadas 1000 iterações do ciclo?

(c) [2 valores] Altere o código apresentado, reordenando as instruções e, se considerar útil, modificando o *offset* das instruções de acesso à memória, de modo a eliminar o maior número possível de atrasos e de ciclos desperdiçados durante a sua execução no *pipeline* com *forwarding*, com decisão dos saltos condicionais no andar ID e com um *branch delay slot*.

9. [3 valores] As latências das várias componentes do *pipeline* são as apresentadas na tabela seguinte:

PC	Registos do *pipeline*	Memória	Banco registos	ALU	Somadores	*Multiplexors*	Controlo	Controlo da ALU
10 ps	15 ps	350 ps	200 ps	330 ps	250 ps	40 ps	40 ps	20 ps

Considere que os restantes elementos lógicos têm latência zero.

Quais as unidades funcionais, de controlo, etc., que se encontram no caminho crítico de cada andar do *pipeline*? Qual a latência desses caminhos?

Qual o andar que determina o período mínimo do ciclo de relógio? Qual é esse período?

Nome: _____ Número: _____

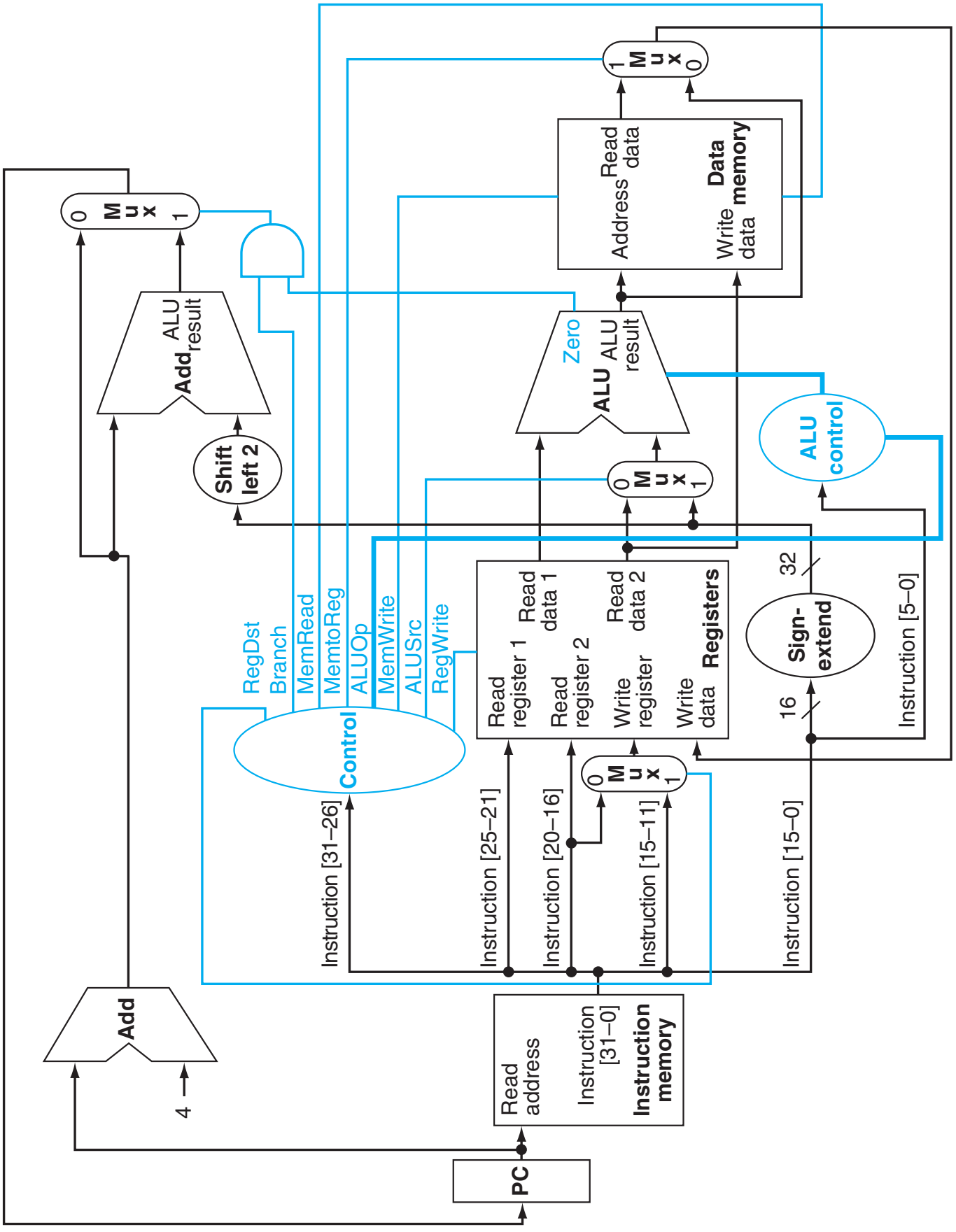

Figura 1: Diagrama de blocos da implementação MIPS monociclo

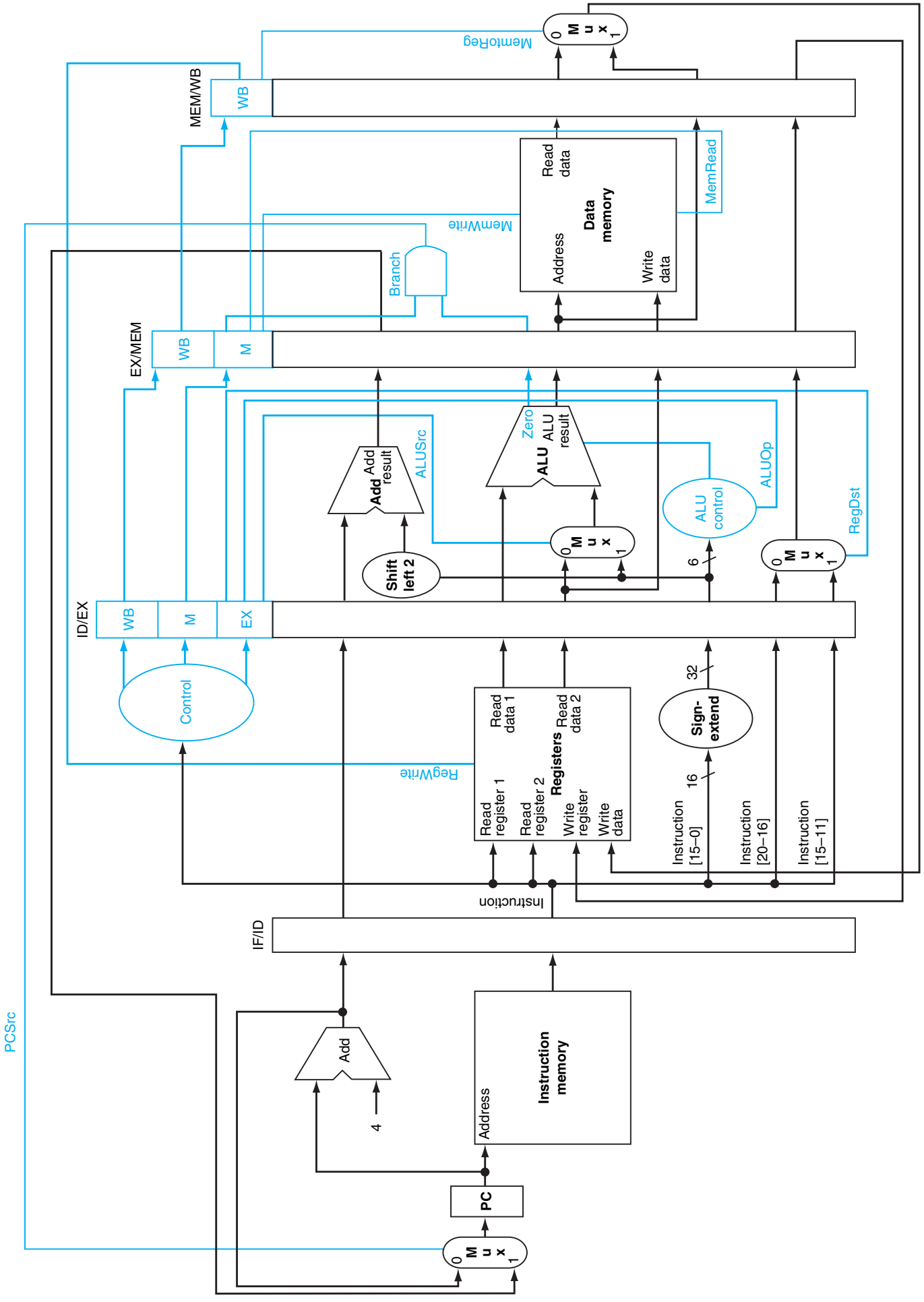

Figura 2: Diagrama de blocos do *pipeline* MIPS