Arquitectura de Sistemas e Computadores II
Exame de Recurso

Departamento de Informática
Universidade de Évora

23 de Janeiro de 2017

Indique todos os cálculos efectuados

Perguntas rápidas

1. [0,5 valores] Sabendo que um programa executa menos instruções no processador X do que as executadas pelo programa equivalente no processador Y, pode concluir que o desempenho de X é superior ao de Y?

2. [0,5 valores] Nos processadores reais, a que correspondem as duas memórias distintas, de instruções e de dados, visíveis nos diagramas de blocos do MIPS?

3. [0,5 valores] Numa cache 16-*way set associative* com 512 conjuntos, em quantas posições pode ser colocado um dado bloco?

4. [0,5 valores] Qual a relação entre a dimensão das páginas virtuais e a das páginas físicas de um sistema?

Desempenho

5. [2 valores] Qual o período do relógio de um computador em que a execução de 4000 milhões de instruções, distribuídas pelas diferentes classes de acordo com a tabela abaixo, demora 7,4 s?

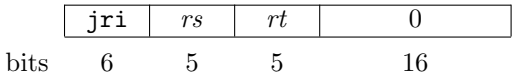
Classe	Aritméticas	Acesso à memória	Saltos
%	50	35	15
CPI	2	6	4

6. [2 valores] Qual o *speedup* que se obtém tornando 4 vezes mais rápida a execução das instruções da classe responsável por 80% do tempo de execução de um programa?

Implementação MIPS monociclo

7. [4 valores] Pretende-se que a implementação MIPS monociclo da Figura 1 suporte a execução da instrução `jri` (*jump register indexed*), que é uma instrução tipo-R com dois argumentos:

`jri rs, rt`



Esta instrução provoca a continuação da execução na instrução cujo endereço é obtido somando os conteúdos dos registos `rs` e `rt`.

(a) Quais das unidades funcionais e dos *multiplexers* existentes serão usados na execução desta instrução?

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

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

(d) Apresente na Figura 1 as alterações à implementação que considerar necessário fazer.

(CONTINUA. . .)

Pipeline MIPS de 5 andares

8. [2 valores] Simule a execução do código à direita, num processador com *forwarding*, com decisão dos saltos condicionais no andar ID, com previsão perfeita do resultado das instruções de salto condicional e sem *delay slots*, assumindo que o valor no registo `$t1` é 1 na primeira vez que a instrução 6 é executada. 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.

```
1.   inicio: lw    $t0, 0($a0)
2.           add   $t0, $t0, $t0
3.           addu  $t0, $t0, $t0
4.           slt   $t1, $a1, $t0
5.           addiu $a0, $a0, 4
6.           beq   $t1, $0, inicio
7.   fim:    addi  $v0, $a0, -4
8.           jr    $ra
```

Quantos ciclos de relógio seriam necessários para executar o código se o ciclo (instruções 1 a 6) fosse executado 200 vezes?

Cache

9. [2 valores] Complete o que puder da figura seguinte, que representa parcialmente o estado de uma cache *2-way set associative*, com 4 conjuntos e blocos de 2 palavras, de um processador com palavras de 64 bits. (Na figura, M[p] representa o valor da palavra de memória p.)

Índice	*Valid*	*Tag*	Palavras		*Valid*	*Tag*	Palavras	
0	0				0			
1		5			1	5		
2	0				0			
3	1		M[46]		0			

Memória virtual

10. Considere um sistema MIPS em que a dimensão das páginas de memória é de 4KB. Num momento da execução de um programa, a sua tabela de páginas apresenta o conteúdo (parcialmente) mostrado abaixo.

Tabela de páginas

	Dirty	Pág. física
	...	
10	1	17
11	0	25
12	1	9
13	0	DISCO
14	0	30
	...	

(a) [2 valores] Assumindo que as páginas virtuais foram acedidas pela ordem 10, 11, 12, 10 e 14, apresente o conteúdo do TLB do sistema (2-*way set associative*, com 4 blocos de uma tradução e substituição por LRU).

(b) [2 valores] Calcule o endereço físico correspondente ao endereço virtual $00\,B9\,E8_{16}$.

Multiprocessamento

11. [2 valores] Pretende-se implementar um mecanismo para a sincronização de 2 *threads* de um processo num sistema multiprocessador MIPS de memória partilhada, que permita garantir que nenhuma das *threads* continua a execução para lá de um ponto de sincronização até ambas terem alcançado esse ponto. A base desse mecanismo é a função `barreira`, que todas as *threads* devem invocar no ponto de sincronização.

As versões C e MIPS (sem *delay slots*) da implementação proposta para a função são apresentadas abaixo. O valor inicial da variável (partilhada) `faltam` é 2. O objectivo é que a variável tome o valor zero depois de ambas as *threads* invocarem a função, e só nessa altura, para as *threads* poderem prosseguir.

```
void barreira()
{
  faltam = faltam - 1;

  while (faltam != 0)
    ;
}
```

```
barreira: lw $t0, faltam      # lê valor de faltam
          addiu $t0, $t0, -1
          sw $t0, faltam      # actualiza o valor
ciclo:    lw $t0, faltam      # relê valor de faltam
          bne $t0, $0, ciclo
fim:      jr $ra
```

A versão MIPS da implementação da função garante:

i. que as 2 *threads* só poderão prosseguir depois de ambas atingirem o ponto de sincronização?

ii. e que, tendo ambas atingido esse ponto, ambas poderão eventualmente prosseguir?

Justifique as suas respostas.

Nome: _____ Número: _____

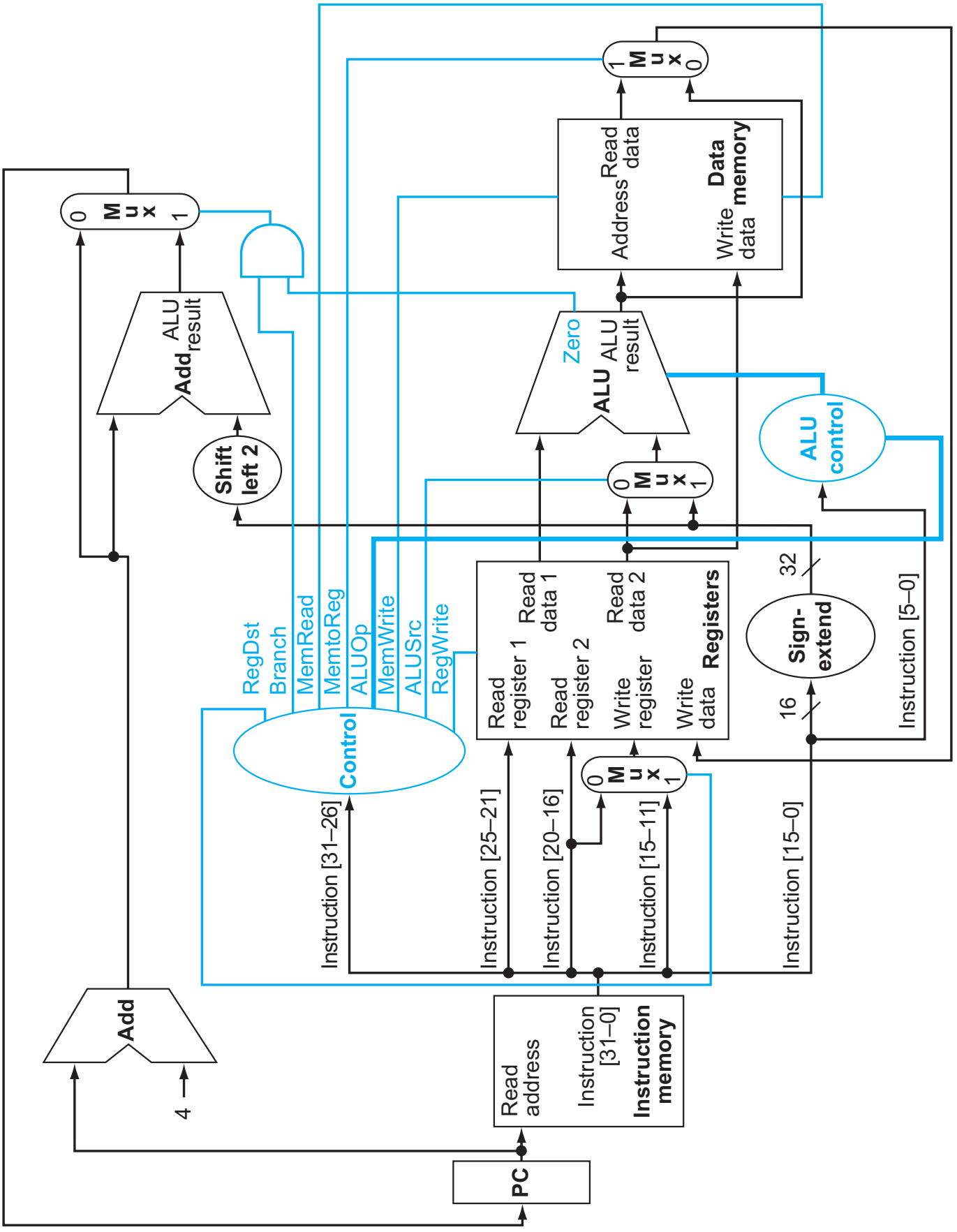


Figura 1: Diagrama de blocos da implementação MIPS monociclo

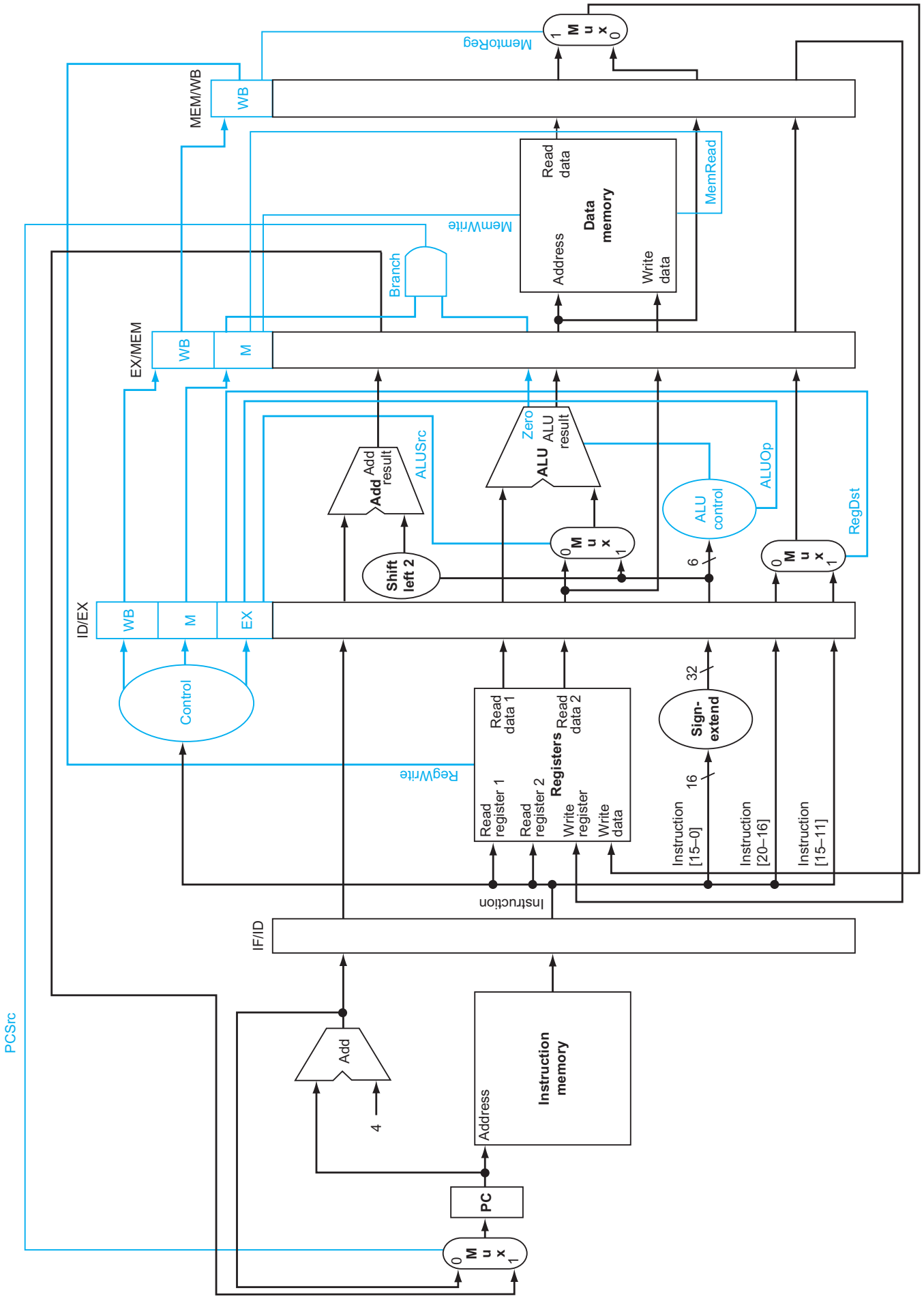


Figura 2: Diagrama de blocos do *pipeline* MIPS