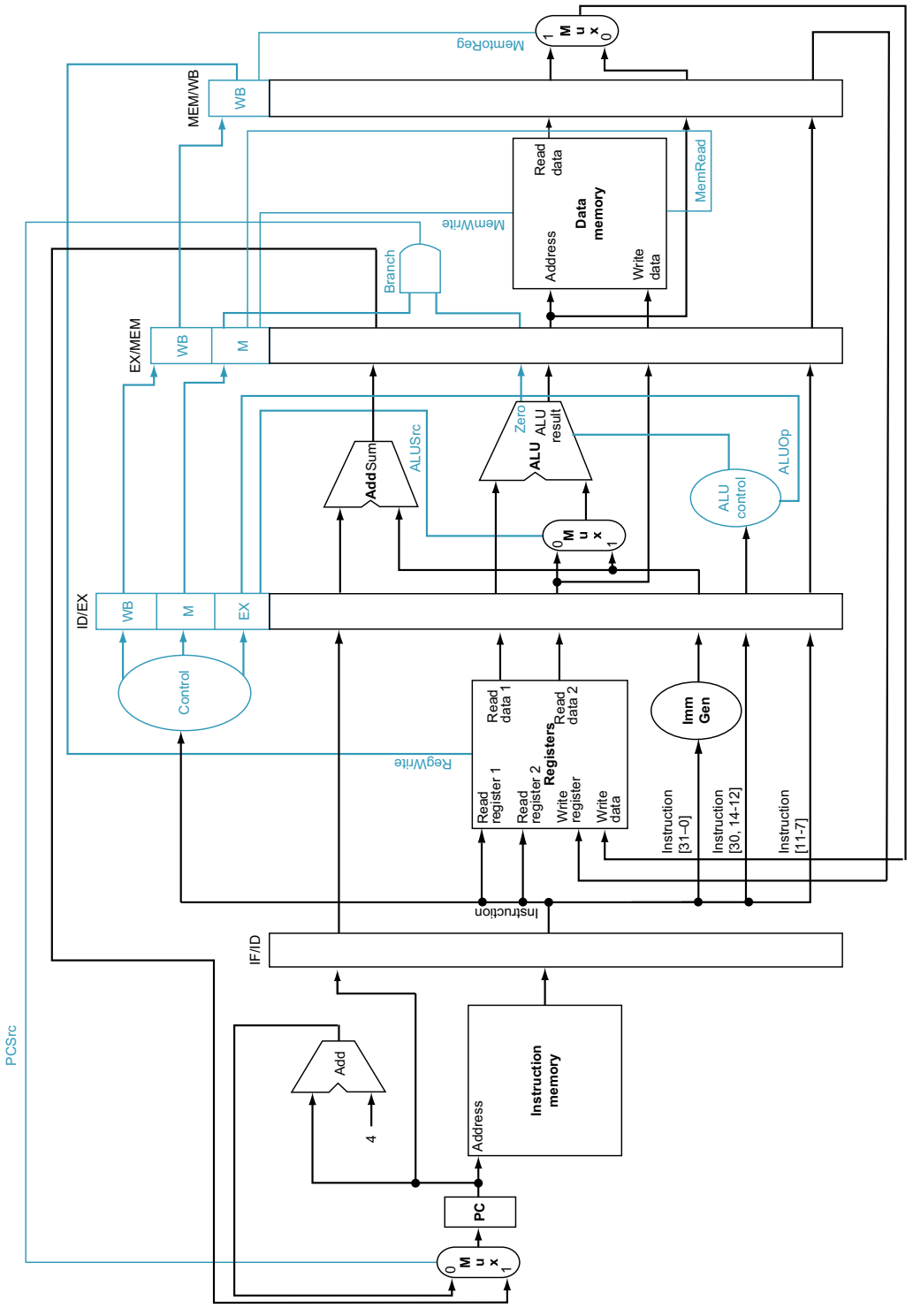
Arquitetura de Computadores II
3.ª Frequência / Exame

Departamento de Informática / Universidade de Évora

4 de janeiro de 2024

Identifique todas as folhas. Responda e justifique as respostas nas caixas de texto.



Perguntas rápidas

1. [1 valor] Um programa P é posto em execução em dois processadores X e Y, da mesma arquitetura, com as características indicadas na tabela. Qual o processador com maior desempenho e qual o *speedup* em comparação com o outro de menor desempenho?

	X	Y
Frequência de relógio	1 GHz	800 MHz
CPI médio	1.5	1.3

2. [1 valor] Um processador, com frequência de relógio de 2 GHz, executa 10^6 instruções de três classes diferentes em 1.42 ms. Preencha os espaços em falta na tabela abaixo:

Classe	X	Y	Z
Instruções	120000		600000
CPI	4	2	

3. [1 valor] Quais os motivos que dão origem a um CPI maior que 1 numa implementação *pipelined*?

4. [1 valor] Considere uma implementação RISC-V *2-way static multiple issue*, em que cada *issue packet* pode conter uma instrução aritmética ou salto, e uma instrução de acesso à memória. Se 20% das instruções de um programa forem de acesso à memória, qual o *speedup* que se obtém, numa situação ideal, face a uma implementação *pipelined* simples?

5. [1 valor] Um processador da arquitetura RISC-V tem uma *cache direct mapped* de 32 KiB com blocos de 4 words. Quantas linhas tem a *cache*?

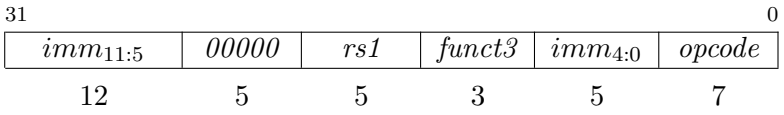
6. [0.5 valores] Se, ao traduzir um endereço virtual para endereço físico, o número de página físico não se encontrar na tabela de páginas, obtém-se um: ◯ *Miss.* ◯ *Page fault.* ◯ *Segmentation fault.*

7. [1 valor] Se um endereço virtual tiver 36 bits, uma página tiver 8 KiB, 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?

RISC-V — implementação *pipeline* de 5 andares

8. [3 valores] Pretende-se que a implementação RISC-V *pipelined* suporte a execução da instrução `spc` (*store program counter*), que é uma instrução do tipo S com dois argumentos:

`spc imm(rs1)`



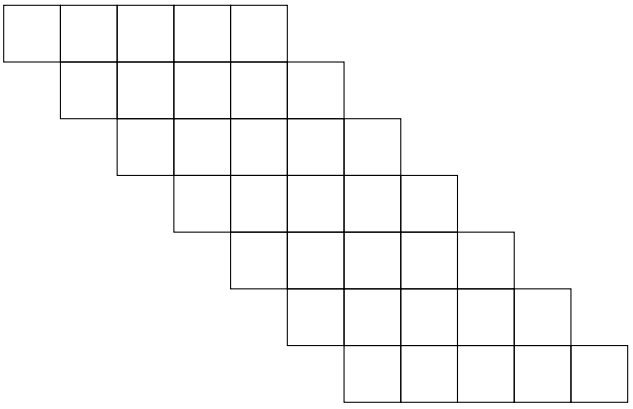
Esta instrução guarda o valor atual do *program counter*, em memória, no endereço calculado de modo semelhante às instruções de *store*, somando o *immediate* ao registo `rs1`. Que unidades funcionais (incluindo multiplexers) e que sinais de controlo é necessário acrescentar? Faça as alterações necessárias diretamente no diagrama de blocos da figura e preencha a seguinte tabela com os sinais de controlo. Se for necessário adicionar um sinal de controlo extra, use a coluna vazia da tabela.

ALUSrc	PCSrc	ALUOp	RegWrite	MemRead	MemWrite	MemToReg	

9. [0.5 valores] A execução de um programa num processador com execução fora de ordem pode levar a um resultado diferente do obtido na execução do mesmo programa num processador em que as instruções são executadas estritamente por ordem? ◯ Sim ◯ Não

10. Considere o código RISC-V seguinte.

```
lui x5, 0x10004     # x5 <-
addi x5, x5, 0x3ac  # x5 <-
lw x6, 0(x5)        # x5 <-       x6 <-
srli x7, x6, 1      # x6 <-       x7 <-
addi x5, x5, 4      # x5 <-
sw x6, 0(x5)        # x5 <-       x6 <-
```



`x1 <- 3` indica uma dependência do valor escrito em `x1` há 3 instruções atrás (1 significa instrução anterior). Deixe em branco se não houver dependência.

(a) [1 valor] Indique todas as dependências de dados nos espaços reservados em comentário.

(b) [1 valor] Assinale, rodeando com um rectangulo, as dependências que dão origem a conflitos de dados. Por exemplo, `x1 <- 3`.

(c) [2 valores] Simule a execução destas instruções no pipeline RISC-V, com *forwarding*. 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.

11. Considere um sistema com palavras e endereços de 32 bits, uma *cache 2-way set associative* com 512 conjuntos, blocos de 8 bytes e estratégia de substituição LRU. A figura mostra apenas as primeiras 4 linhas da *cache* (índices 0 a 3).

V	D	Tag	Data	V	D	Tag	Data

Hit	Miss	Endereço
◯	◯	`0x10040004`
◯	◯	`0x10040008`
◯	◯	`0x10040000`
◯	◯	`0x1004000c`
◯	◯	`0x10042004`
◯	◯	`0x1006a000`

(a) [1 valor] Qual a capacidade desta *cache* para guardar dados?

(b) [1 valor] Suponha que a *cache* está inicialmente vazia quando se começam a fazer, sequencialmente, acessos a dados nos endereços indicados. Mostre o estado final da cache após estes acessos. Indique também, para cada acesso, se ocorreu um *hit* ou *miss*.

(c) [1 valor] Calcule o *hit rate* e o *miss rate*. Se o *hit time* for de 2 ciclos de relógio e o *miss penalty* 50 ciclos, qual o tempo médio de acesso à memória? (se não respondeu à pergunta anterior, escolha um *hit rate* entre 10% e 90%)

12. [3 valores] A tabela de páginas, cujo conteúdo é parcialmente mostrado na tabela da esquerda, pertence a um processo que corre num processador com um TLB *direct mapped*, com capacidade para 4 traduções, e estratégia *write-through*.

Assuma que o TLB inicialmente está vazio. Para cada operação de tradução (acessos indicados na tabela da direita), atualize o conteúdo da TLB, indique se ocorreu um *hit* ou *miss* e, se aplicável, a página virtual cuja tradução será substituída.

Tabela de páginas

Index	Dirty	PPN
	...	
4	0	11
5	0	73
6	1	39
7	0	22
8	0	9
9	0	55
10	0	91
11	1	17
	...	

TLB

V	Dirty	Tag	Physical Page Number

Hit	Miss	Acesso	Virtual Page Number
◯	◯	Leitura	9
◯	◯	Leitura	11
◯	◯	Escrita	11
◯	◯	Leitura	7
◯	◯	Escrita	5