Prova Teórica

Arquitetura e Organização de Computadores II

Prof. Roberto Cabral

2º Semestre de 2021

Aluno: [ ] Matrícula: [ ]

1. Considerando o conteúdo dos registradores mostrados abaixo, elabore uma tabela, para cada item, que represente o estado da memória (endereço/conteúdo) após a execução das seguintes instruções:

```
R10 = 0x55aabb44    R11 = 0x00001234    R12 = 0x00001000  R13 = 0x2000e000
```

   (a) `stmfd sp!, r10-r12`

   (b) `stmea sp!, r10`

   (c) `stmib r10!, r11,r12`

   (d) `stmia sp!, r11,r12`

   (e) `str r10, [r12]`

   (f) `str r11, [r10, r12]!`

   (g) `str r10, [r1], #4`

   (h) `stmed sp!, r10-r12`

2. Explique por que a arquitetura ARM não é puramente RISC.

3. O que acontece quando ocorre uma interrupção no ARM?

4. Explique o funcionamento do pipeline do ARM.

5. Diferencie os seguintes modos de endereçamento: *Preindex with writeback*, *Preindex* e *Postindex*.

6. Explique o propósito do SPSR, e dê uma justificativa do para o fato dele não estar disponível nos modos USER e SYSTEM.

7. Explique o que é e como funciona o *Translation Lookaside Buffer* em uma MMU.

8. Supondo que um cache tenha as seguintes características:

   - 1024 linhas;
   - 128 bits por linha;
   - 32 bits de tamanho de endereço do processador;
   - 4 conjuntos associativos.

   Responda:

   (a) Qual o tamanho em bytes da memória cache. Justifique.

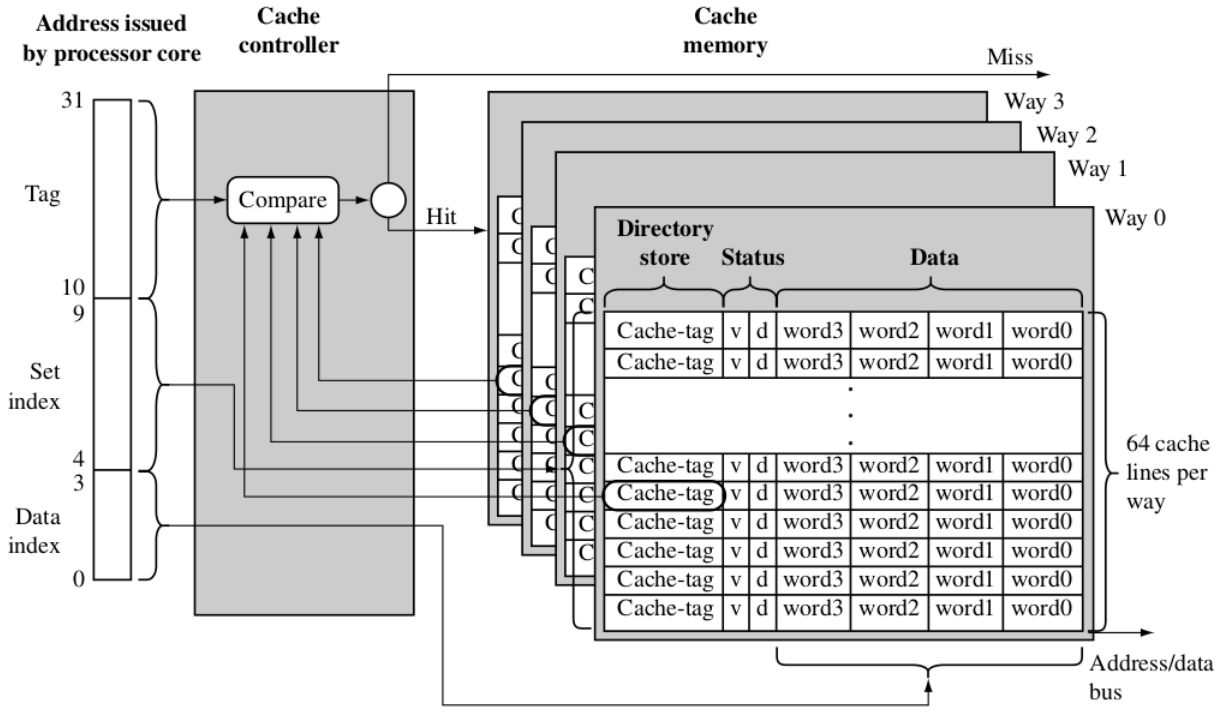
   (b) Quais bits do endereço são responsáveis pelo endereçamento do byte dentro do bloco Justifique.

   (c) Quais bits do endereço determinam uma linha do cache? Justifique.

   (d) Quais bits do endereço são responsáveis pela *tag*? Justifique.

   (e) Liste 4 endereços não consecutivos que façam parte de uma mesma linha do cache e ocupem diferentes conjuntos associativos. Justifique.

9. Explique o funcionamento da cache abaixo:



10. Considerando cada pré-condição, indique uma instrução ARM que resulte na respectiva pós-condição, dada a memória abaixo.

Memória

```
mem32[0x80020] = 0x00000005
mem32[0x8001C] = 0x00000004
mem32[0x80018] = 0x00000003
mem32[0x80014] = 0x00000002
mem32[0x80010] = 0x00000001
mem32[0x8000C] = 0x00000000
```

Pré-condição	**Pré-condição**	**Pré-condição**	**Pré-condição**
r0 = 0x00080010	r0 = 0x00080010	r0 = 0x0008001C	r0 = 0x00080014
r1 = 0x00000000	r1 = 0x00000000	r1 = 0x00000002	r1 = 0x00000001
r2 = 0x00000000	r2 = 0x00000000	r2 = 0x00000003	r2 = 0x00000002
r3 = 0x00000000	r3 = 0x00000000	r3 = 0x00000004	r3 = 0x00000003
Instrução			
---	---	---	---
Pós-condição	**Pós-condição**	**Pós-condição**	**Pós-condição**
r0 = 0x0008001C	r0 = 0x0008001C	r0 = 0x00080010	r0 = 0x00080020
r1 = 0x00000001	r1 = 0x00000002	mem32[0x80010] = 0x02	mem32[0x8001C] = 0x03
r2 = 0x00000002	r2 = 0x00000003	mem32[0x80014] = 0x03	mem32[0x80018] = 0x02
r3 = 0x00000003	r3 = 0x00000004	mem32[0x80018] = 0x04	mem32[0x80014] = 0x01