Exercício 1

Responda às perguntas seguintes:

a. Quantos são os valores possíveis de um bit e quais são eles?

b. Quantos bits tem um *byte*?

c. Quantos *bytes* existem em 32 bits? E em 64?

d. Quantos *bytes* existem em 1 KiB? E em 1 MiB? E em 1 GiB?

e. O que é um endereço?

f. Quantos *bytes* é possível endereçar com endereços de 4 bits?

g. Quantos *bytes* é possível endereçar com endereços de 32 bits? A quantos KiB correspondem? E a quantos MiB? E a quantos GiB?

h. Quantos *bytes* é possível endereçar com endereços de 64 bits? A quantos GiB correspondem? E a quantos TiB? E a quantos PiB? E a quantos EiB?

Exercício 2

Responda às seguintes perguntas sobre a arquitectura RISC-V (32 bits):

a. Quantos registos de uso genérico tem um processador que implementa a arquitectura RISC-V?

b. Onde se localizam os registos RISC-V?

c. Qual a capacidade de um registo, em bits? E em palavras?

d. Se o registo `t0` contiver o valor 10000001_{16}, diga quais das seguintes instruções são válidas:

   i. `lw t1, 0(t0)`
   ii. `lw t1, 3(t0)`
   iii. `sw t1, -1(t0)`

e. Qual é o endereço da primeira palavra cujo endereço é não inferior a 10000001_{16}?

Exercício 3

Interprete o código seguinte e descreva o que ele calcula. Assuma que a0 % 4 = 0, a1 > 0, e que o resultado é o valor final em a0:

```
        lw    t0, 0(a0)
ciclo:  addi  a1, a1, -1
        beq   a1, zero, fim
        addi  a0, a0, 4
        lw    t1, 0(a0)
        slt   t2, t0, t1
        beq   t2, zero, ciclo
        or    t0, t1, t1
        jal   zero, ciclo
fim:    ori   a0, t0, zero
        jalr  zero, 0(ra)
```

Exercício 4

a. Escreva código RISC-V para uma função que calcula e devolve a soma dos valores de um vector de inteiros.

   Argumentos da função:

`a0`	endereço do vector
`a1`	dimensão do vector (≥ 0)

b. Quantas instruções serão executadas na função que escreveu se a dimensão do vector for 1000?

Exercício 4

Código

```
        or    t0, zero, zero
        beq   a1, zero, fim
ciclo:  lw    t1, 0(a0)
        add   t0, t0, t1
        addi  a1, a1, -1
        addi  a0, a0, 4
        bne   a1, zero, ciclo
fim:    or    a0, t0, zero
        jalr  zero, 0(ra)                 # return
```

Exercício 5

Sem recorrer a pseudo-instruções, faça uma implementação RISC-V da seguinte função, em que o argumento da função é passado em a0 e o resultado é devolvido em a0:

```
int even(int n)
{
  if (n == 0)
    return 1;

  return odd(n - 1);
}
```

Exercício 6

Sem recorrer a pseudo-instruções, faça uma implementação RISC-V recursiva da seguinte função, em que o argumento da função é passado em a0 e o resultado é devolvido em a0:

```
int zigzag(int n)
{
  if (n == 0)
    return 0;

  return n - zigzag(n - 1);
}
```

Exercício 1.5 (CODV)

Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

c. We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

Exercício 1.6 (CODV)

Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz CPIs of 2, 2, 2, and 2.

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which implementation is faster?

a. What is the global CPI for each implementation?

b. Find the clock cycles required in both cases.

c. Which implementation is faster?

Exercício 7

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?

Exercício 8

O programa P é executado num processador cujo relógio tem uma frequência de 1 GHz. A sua execução demora 4 s e corresponde à execução de 2000 milhões de instruções.

a. Quantos milhões de instruções são executados por segundo?

b. Qual o CPI de P?

c. O programa Q é outra versão do mesmo programa, para a qual, no mesmo processador, são executadas 3000 milhões de instruções, com um CPI de 1,6. Quanto tempo demora a sua execução?

d. Na execução de Q, quantos milhões de instruções são executadas por segundo?

Exercício 1.7 (CODV)

Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.

b. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

c. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

Exercício 9

Considere a implementação (parcial) RISC-V dada na teórica.

a. Escreva, em binário, a instrução `sub x7, x6, x17`. (O *opcode* da instrução é 51, o campo *funct7* tem o valor 32, e o campo *funct3* tem o valor 0.)

b. Considerando que o endereço da posição em que encontra a instrução é o 400008_{16} e que o valor nos registos 6 e 17 é, respectivamente, 17902 e 19, indique o valor presente em cada linha do circuito no fim da execução da instrução.

c. Considere os seguintes tempos de resposta das unidades funcionais. (Assuma que toda a lógica não mencionada na tabela tem tempo de resposta 0 s e que os valores dos sinais de controlo ficam disponíveis no instante que em que termina a leitura da instrução.)

PC	Mem instr.	Somador	*Muxes*	Banco de registos	ALU	Mem dados	*Imm Gen*
25 ps	200 ps	70 ps	20 ps	80 ps	90 ps	250 ps	15 ps

Se a escrita do endereço da instrução no PC se iniciar no instante 0, quanto tempo é necessário até que os valores que indicou estejam disponíveis?

Exercício 4.1 (CODV)

Considere a instrução

```
and rd, rs1, rs2
```

que coloca no registo `rd` o valor de `rs1 AND rs2`, calculado bit a bit.

a. Quais os valores dos sinais de controlo durante a execução desta instrução e qual a operação realizada pela ALU? (Não é necessário determinar o valor de `ALU operation`.)

b. Quais as unidades funcionais (incluindo *multiplexers*) que têm um papel útil no funcionamento desta instrução?

c. Quais as unidades funcionais (incluindo *multiplexers*) que, durante o funcionamento da instrução, produzem valores que não são usados? E quais não produzem valor?

Exercício 10 (1)

A implementação da figura considera um conjunto limitado de instruções RISC-V. É sempre possível acrescentar instruções a uma arquitectura, mas essa decisão depende das implicações a nível da complexidade e do custo acrescidos do caminho de dados e do controlo que a sua inclusão acarreta.

Neste exercício, considere a nova instrução:

```
lwi rd, rs2(rs1)
```

que coloca no registo `rd` a palavra que se encontra na memória, a partir da posição cujo endereço é `rs1 + rs2`.

a. Quais das unidades funcionais (incluindo os *multiplexers*) existentes podem ser usados para esta instrução?

b. Que unidades funcionais (incluindo *multiplexers*) é necessário acrescentar para implementar esta instrução?

c. Que novos sinais de controlo são necessários para implementar esta instrução?

Exercício 10 (2)

d. Quais os valores de todos os sinais de controlo durante a execução desta instrução e qual a operação realizada pela ALU? (Não é necessário determinar o valor de `ALU operation`.)

Faça na figura as alterações que considerar necessárias para a implementação da instrução.

Repita o exercício para a nova instrução:

```
seq rd, rs1, rs2
```

que coloca no registo `rd` o valor 1 se o valor contido em `rs1` é igual ao contido em `rs2`, e o valor 0 se são diferentes.

(Este exercício deverá ser resolvido sem alterar a ALU, que só realiza as operações AND, OR, soma e subtracção.)

Exercício 11

Pretende-se modificar a implementação monociclo do RISC-V da Figura 4.21 para suportar a instrução `jal`, que é uma instrução tipo-UJ.

a. Quais das unidades funcionais (incluindo os *multiplexers*) existentes serão usadas na execução da instrução?

b. Que unidades funcionais (incluindo *multiplexers*) é necessário acrescentar?

c. Que novos sinais de controlo são necessários?

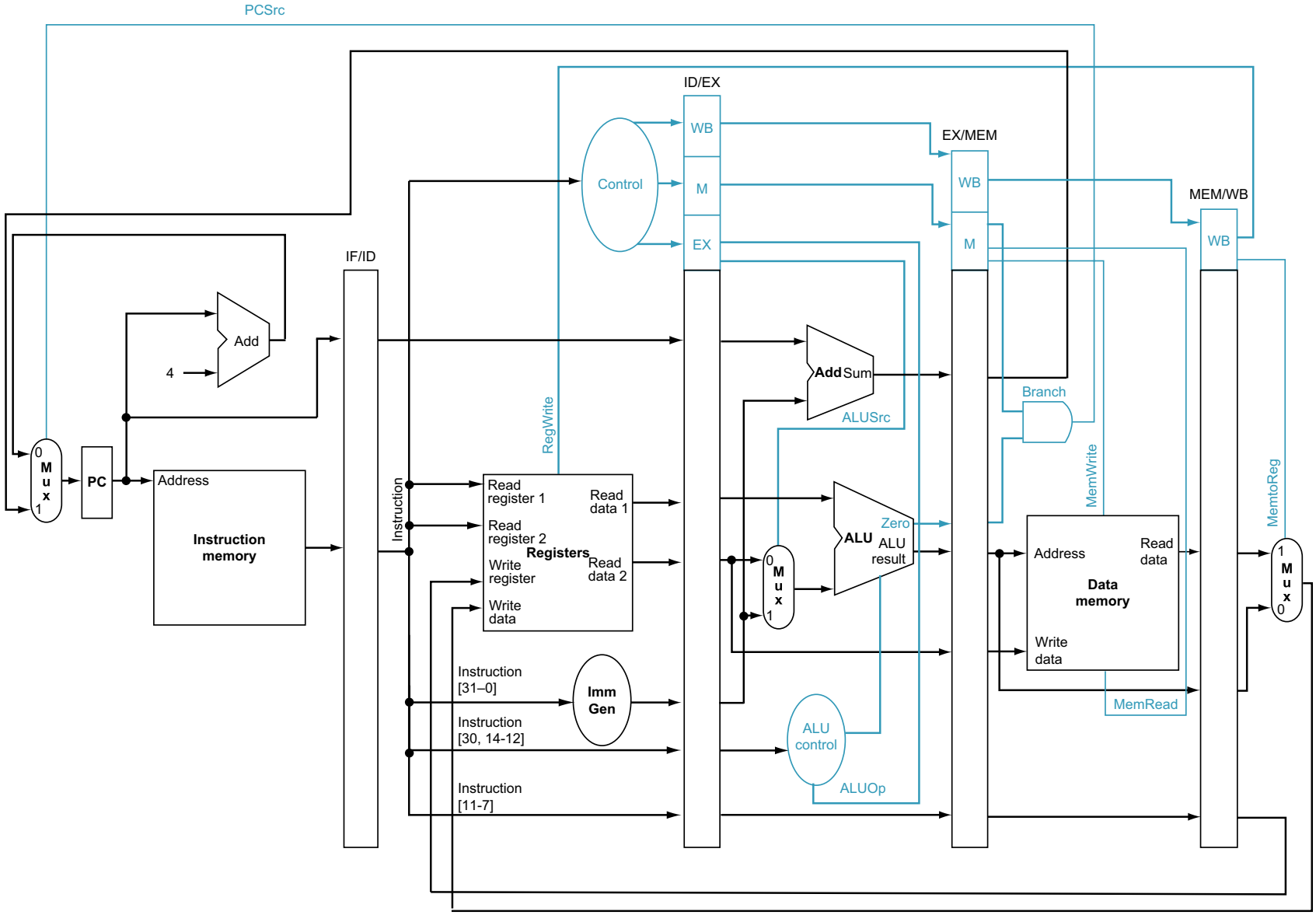
d. Quais os valores de todos os sinais de controlo durante a execução desta instrução e qual a operação realizada pela ALU? (Não é necessário determinar o valor de ALUOp.)

Faça na Figura 4.21 as alterações que considerar necessárias para a implementação da instrução.

Repita o exercício para as instruções:

- `bne`
- `jalr`
- `...`

Figura 4.53


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
Instruction [30, 14-12]
Instruction [11-7]
Imm Gen
Add
Sum
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
Branch
MemWrite
MemRead
MemtoReg
Address
Read data
Data memory
Write data

Exercício 4.16 (CODV2)

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

ALU/Logic	Jump/Branch	Load	Store
45%	20%	20%	15%

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an `lw` instruction in a pipelined and non-pipelined processor?
3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?
4. Assuming there are no stalls or hazards, what is the utilization of the data memory?
5. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

Exercício 12

Calcule o CPI para a implementação *pipelined* do RISC-V:

a. para 1 instrução;

b. para um milhão de instruções;

c. para um milhão de instruções quando 1 em cada 5 é atrasada um ciclo.

Exercício 13

Considere as seguintes latências:

Registos do *pipeline*	Somador do Branch	*Multiplexers*	ALU	Controlo da ALU
40 ps	150 ps	30 ps	200 ps	25 ps

Calcule o caminho crítico do andar EX da implementação RISC-V *pipelined* (Figura 4.53 do CODV2). Diga qual a sua duração e quais as unidades funcionais que o compõem.

Exercício 14

Na resolução deste exercício, assuma que a decisão sobre se um salto condicional é ou não efectuado é tomada no andar MEM (ver Figura 4.53) e que não há qualquer atraso do *pipeline* devido à decisão dos saltos condicionais.

Para o código abaixo:

```
1.              or   x3, x0, x0
2.              or   x8, x5, x0
3.      início: beq  x8, x0, fim
4.              addi x4, x4, -4
5.              lw   x9, 0(x4)
6.              add  x3, x3, x9
7.              addi x8, x8, -1
8.              beq  x0, x0, início
9.      fim:    sub  x3, x7, x3
```

a. Identifique todas as dependências (de dados) existentes. (Cuidado com os saltos.)

b. Identifique os conflitos de dados.

c. Introduza os `nop` necessários para eliminar os conflitos num *pipeline* sem *forwarding*.

d. Introduza os `nop` necessários para eliminar atrasos no *pipeline* com *forwarding*.

Exercício 14 (1)

Na resolução deste exercício, assuma que a decisão sobre se um salto condicional é ou não efectuado é tomada no andar MEM (ver Figura 4.53) e que não há qualquer atraso do *pipeline* devido à decisão dos saltos condicionais (há previsão perfeita do efeito dos saltos condicionais).

Para o código abaixo:

```
1.              or   x3, x0, x0
2.              or   x8, x5, x0
3.      início: beq  x8, x0, fim
4.              addi x4, x4, -4
5.              lw   x9, 0(x4)
6.              add  x3, x3, x9
7.              addi x8, x8, -1
8.              beq  x0, x0, início
9.      fim:    sub  x3, x7, x3
```

e. Apresente a evolução do estado do *pipeline* com *forwarding* durante a execução do código, sendo que o teste da instrução 3 sucede da segunda vez que a instrução é executada. Assinale todos os atrasos introduzidos e todos os pontos onde foi necessário fazer *forwarding* de algum valor.

Exercício 14 (2)

f. Considerando somente os ciclos 5 a 11 da simulação feita na alínea anterior, calcule a percentagem dos ciclos de relógio em que todos os andares do *pipeline* estão ocupados por alguma instrução.

g. Se fossem executadas 1000 iterações do ciclo (instruções 3 a 8), qual o número total de instruções executadas? Qual o CPI correspondente?

h. Para o ciclo da execução em que o conteúdo do *pipeline* é o indicado abaixo, qual é o valor dos sinais de controlo em uso em cada um dos andares?

Andar	**Instrução**
IF	`add  x3, x3, x9`
ID	`lw   x9, 0(x4)`
EX	`addi x4, x4, -4`
MEM	`beq  x8, x0, fim`
WB	`or   x8, x5, x0`

Exercício 14 (3)

i. Modifique o código de modo a que, na presença de *forwarding*, a sua execução se possa dar sem a introdução de qualquer atraso.

j. Como lida o processador RISC-V (em termos de atrasos e de *forwardings*) com a execução das duas instruções seguintes, com a decisão dos saltos condicionais a acontecer no andar ID?

```
or   x8, x5, x0
beq  x8, x0, fim
```

Exercício 17 (1)

Considere um *pipeline* RISC-V com *double issue* estático, com decisão dos saltos condicionais no andar MEM, previsão perfeita de saltos condicionais e com *forwarding*.

a. Assuma que o *issue packet* pode conter quaisquer duas instruções, que qualquer instrução do *issue packet* pode ser atrasada em relação à anterior (o que provoca o atraso de *todas* as instruções que se seguem), que pode haver *forwarding* da 1ª para a 2ª instrução do *issue packet*, e que entram sempre duas instruções, em cada ciclo, no *pipeline*.

```
 1.       or   x5, x0, x0
 2. for:  slli x10, x5, 2
 3.       beq  x5, x6, fim
 4.       add  x11, x1, x10
 5.       lw   x12, 0(x11)
 6.       lw   x13, 4(x11)
 7.       sub  x12, x12, x13
 8.       add  x14, x2, x10
 9.       sw   x12, 0(x14)
10.       addi x5, x5, 2
11.       beq  x0, x0, for
12. fim:
```

Simule a execução do código (sequencial) acima, para uma iteração do ciclo (até à 2ª vez que a instrução da linha 3 é executada), apresentando a evolução do conteúdo do *pipeline* e os *forwardings* efectuados.

Nota: Este código RISC-V corresponde ao código C seguinte:

```
for (i = 0; i != j; i += 2)
  b[i] = a[i] - a[i + 1];
```

Exercício 17 (2)

Os registos atribuídos às várias variáveis foram os seguintes:

a	b	i	j
x1	x2	x5	x6

b. Organize o código apresentado para ser executado no *pipeline* RISC-V com *double issue* estático (com decisão dos saltos no andar ID, previsão perfeita, e sem *forwarding* entre as duas instruções do *issue packet*), em que cada *issue packet* pode conter uma instrução aritmética ou de salto e uma instrução de acesso à memória, de modo a não haver a necessidade da introdução de atrasos durante a sua execução.

c. Calcule o IPC verificado em cada uma das alíneas anteriores, para uma iteração do ciclo.

d. Assumindo que `j` é sempre um múltiplo de 4, desdobre o ciclo 2 vezes e organize o código para ser executado nas condições da alínea b. (Se necessitar de usar registos adicionais, pode usar os registos `x15` e `x16`.)

   Calcule o IPC verificado em *cada iteração* do novo ciclo e compare-o com o verificado em *cada iteração* do ciclo da alínea b.

Exercício 17 (3)

Resolução possível da alínea b:

```
          ALU ou salto                Acesso à memória

          or   x5, x0, x0
for:      slli x10, x5, 2
          beq  x5, x6, fim
          add  x11, x1, x10
          add  x14, x2, x10           lw   x12, 0(x11)
          addi x5, x5, 2              lw   x13, 4(x11)

          sub  x12, x12, x13
          beq  x0, x0, for            sw   x12, 0(x14)
fim:
```

Exercício 17 (4)

Resolução possível da alínea d:

```
                ALU ou salto                Acesso à memória

                or   x5, x0, x0
for:            slli x10, x5, 2
                beq  x5, x6, fim
                add  x11, x1, x10
                add  x14, x2, x10           lw   x12, 0(x11)
                addi x5, x5, 4              lw   x13, 4(x11)
                                            lw   x15, 8(x11)
                sub  x12, x12, x13          lw   x16, 12(x11)
                                            sw   x12, 0(x14)
                sub  x15, x15, x16
                beq  x0, x0, for            sw   x15, 8(x14)
fim:
```

Exercício 18

Para cada uma das instruções do segmento de código RISC-V abaixo, a partir da da linha 2, diga, justificando, se essa instrução poderia ser executada antes da instrução da linha 1, num processador com *pipeline scheduling* dinâmico (ie, com execução de instruções fora de ordem).

```
1.    and  x5, x6, x7
2.    or   x7, x8, x9
3.    add  x10, x5, x7
4.    sw   x12, 0(x13)
5.    lw   x5, 20(x12)
6.    jalr x0, 0(x5)
```

Exercício 19

Simule o funcionamento de uma cache, inicialmente vazia, com colocação (em posições) fixa(s) (*direct-mapped*), com 16 índices e blocos de uma palavra, durante o acesso à sequência de endereços abaixo (em binário), num sistema com palavras de 32 bits e endereços de 16 bits (os 8 bits mais significativos, não mostrados, têm todos o valor 0):

```
00001100 01001000 00001111 00001000
01101100 01001100 00001100 00111001
```

a. Para cada endereço acedido, e mantendo a notação binária, calcule o número da palavra correspondente, o número do bloco a que a palavra pertence, o índice da posição na cache onde será colocado e o *tag*. Indique, para cada acesso, se houve um *hit* ou um *miss* e, quando isso acontecer, qual o bloco que foi substituído na cache.

   Determine e apresente o estado final da cache.

b. Qual a *miss rate* verificada?

c. Qual a capacidade total desta cache, em bits? Que percentagem dessa capacidade é usada para conter a informação lida da memória?

d. Se um acesso à cache leva 1 ciclo de relógio (*hit time*) e a transferência de uma palavra da memória para a cache custa 20 ciclos (*miss penalty*), qual o tempo total necessário para os acessos indicados?

Exercício 20

Simule o funcionamento de uma cache, inicialmente vazia, com colocação numa de 2 posições (2-*way set associative*), com capacidade para 8 blocos de duas palavras, durante o acesso à sequência de endereços apresentada abaixo, num sistema com palavras de 32 bits e endereços de 16 bits:

```
12 72 15 8 108 76 12 57
```

a. Para cada endereço acedido, e sem recorrer à conversão para binário, calcule o número da palavra correspondente, o número do bloco a que a palavra pertence, o índice da posição na cache onde será colocado e o *tag*. Indique, para cada acesso, se houve um *hit* ou um *miss* e, quando isso acontecer, qual o bloco que foi substituído na cache.

   A escolha do bloco a substituir deve incidir sobre aquele que não é acedido há mais tempo (estratégia LRU).

   Determine e apresente o estado final da cache.

b. Qual a *miss rate* verificada?

c. Qual a capacidade total desta cache, em bits? Que percentagem dessa capacidade é usada para conter a informação lida da memória?

Exercício 21

Um sistema com endereços e palavras de 32 bits possui uma cache 2-*way set associative*. Sabendo que os bits 3-0 de um endereço constituem o *offset* no bloco (o número do *byte* no bloco), que os bits 6-4 são usados para indexar a cache e que os bits 31-7 constituem o *tag* do bloco na cache, calcule:

a. O número de conjuntos da cache.

b. O número de posições (blocos) da cache.

c. O número de palavras por bloco.

Exercício 5.1 (CODV2) (1)

Na resolução deste exercício, assuma os tipos seguintes:

```
int A[8000][8000], B[8][8], I, J;
```

Assuma, também, que o primeiro elemento de cada matriz (posições `[0][0]` *e* `(1,1)`*) é o primeiro elemento de um bloco.*

In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 32-bit integer.

```
for (I = 0; I < 8; I++)
    for (J = 0; J < 8000; J++)
       A[I][J] = B[I][0] + A[J][I];
```

1. How many 32-bit integers can be stored in a 16-byte cache block?
2. References to which variables exhibit temporal locality?
3. References to which variables exhibit spatial locality?

Exercício 5.1 (CODV2) (2)

Locality is affected by both the reference order and data layout. The same computation can also be written below in Matlab, which differs from C by storing matrix elements within the same column contiguously in memory.

```
for I = 1 : 8
    for J = 1 : 8000
       A(I,J) = B(I,1) + A(J,I);
    end
end
```

4. How many 16-byte cache blocks are needed to store all 32-bit matrix elements being referenced?
5. References to which variables exhibit temporal locality?
6. References to which variables exhibit spatial locality?

Exercício 5.1 (CODV2) (3)

7. Nestas alíneas, assuma que a cache de dados do sistema é *fully associative*, com 64KB, e estratégia LRU para substituição de blocos, que o *hit time* da cache é de 1 ciclo e que a *miss penalty* é de 20 ciclos.

   a. Quantos blocos cabem na cache?
   b. Considerando só os acessos a dados, quantos *misses* ocorrerão, e quais serão a *miss rate* e o tempo médio de acesso à memória, na execução da versão C do código?

      Assuma que os valores de `I` e `J`, assim como os endereços das matrizes `A` e `B`, estão em registos do processador.

   c. Se o número total de instruções executadas for 1088051, o CPI base (ie, sem atrasos devido aos acessos à memória) for 1 (consistente com o *hit time* ser de 1 ciclo), e se não ocorrerem *misses* no acesso às instruções, qual o CPI real e quantas vezes mais lento fica o programa devido aos acessos à memória?

Exercício 22 (1)

Na tabela abaixo, são apresentados os tempos de acesso às caches e à memória de um sistema com dois níveis de cache, e as *miss rates* observadas na execução de um programa:

Cache / Memória	*Hit time* / Tempo de acesso	*Miss rate*
L1	1 ciclo	6%
L2	10 ciclos	50%
Memória	100 ciclos	

a. Quanto custa um acesso a um bloco presente na cache L1?

b. Quanto custa um acesso a um bloco presente na cache L2?

c. Quanto custa um acesso a um bloco não presente em cache?

d. Qual é a *miss penalty* para a cache L2?

e. Qual é a *miss penalty* para a cache L1?

Exercício 22 (2)

f. Quanto custa, em média, um acesso à L1?

g. Na ausência da cache L2, quanto custaria, em média, um acesso à cache L1?

h. Quantos dos acessos foram parar à cache L2?

i. Quantos dos acessos foram parar à memória?

j. Qual a *miss rate* global (ie, a *miss rate* combinada das 2 caches)?

k. Se pretendêssemos uma *miss rate* global de 2%, qual deveria ser a *miss rate* da cache L2, se a da cache L1 se mantivesse?

Exercício 23

Considere um sistema com um CPI base de 1 (CPI na ausência de atrasos), com um relógio com uma frequência de 4GHz, com uma memória com uma latência de acesso de 100 ns, e com uma cache em que se verificou, para um programa, uma *miss rate* por instrução de 2%.

a. Qual o CPI real do programa, tendo em conta os atrasos devidos aos acessos à memória?

b. Se for acrescentada uma cache L2 de segundo nível (entre a cache L1 original e a memória), com uma latência de acesso de 5 ns, e essa cache permitir reduzir os acessos à memória para 0.5% (a *miss rate* global por instrução), qual será o *speedup* obtido?

c. Nas condições acima, qual a *miss rate* da cache L2?

Exercício 5.16 (CODV2) (1)

As described in Section 5.7, virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. This exercise shows how this table must be updated as addresses are accessed. The following data constitute a stream of virtual byte addresses as seen on a system. Assume 4 KiB pages, a four-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the [...] largest page number.

`4669, 2227, 13916, 34587, 48870, 12608, 49225`

Exercício 5.16 (CODV2) (2)

`4669, 2227, 13916, 34587, 48870, 12608, 49225`

TLB

Valid	Tag	Physical Page Number	Time Since Last Access
1	11	12	4
1	7	4	1
1	3	6	3
0	4	9	7

Page table

Valid	Physical Page or in Disk
1	5
0	Disk
0	Disk
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12
...	...

Todas as posições da tabela de páginas não mostradas contêm 0 e Disk.

Exercício 5.16 (CODV2) (3)

1. For each access shown above, list
   - whether the access is a hit or miss in the TLB,
   - whether the access is a hit or miss in the page table,
   - whether the access is a page fault,
   - the updated state of the TLB.

   Calcule também o offset na página, o endereço físico correspondente a cada acesso, e a miss rate do TLB. Quando houver substituição de uma tradução do TLB, indique a página virtual correspondente.

   Determine o estado final da tabela de páginas.

2. Repeat Exercise 5.16.1, but this time use 16 KiB pages instead of 4 KiB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

Exercício 5.16 (CODV2) (4)

3. Repeat Exercise 5.16.1, but this time use 4 KiB pages and a [four-entry] two-way set associative TLB.

   Comece com o TLB vazio e preencha-o com o conteúdo correspondente aos acessos às páginas virtuais 11, 3 e 7, por esta ordem.

4. Repeat Exercise 5.16.1, but this time use 4 KiB pages and a [four-entry] direct mapped TLB.

   Comece com o TLB vazio e preencha-o com o conteúdo correspondente aos acessos às páginas virtuais 11, 3 e 7, por esta ordem.

5. Discuss why a CPU must have a TLB for high performance. How would virtual memory accesses be handled if there were no TLB?

Exercício 5.17 (CODV2)

There are several parameters that impact the overall size of the page table. Listed below are key page table parameters.

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	8 KiB	4 bytes

1. Given the parameters shown above, calculate the maximum possible page table size for a system running five processes.
2. Given the parameters shown above, calculate the total page table size for a system running five applications that each utilize half of the virtual memory available, given a two-level page table approach with up to 256 entries at the 1^{st} level. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum amount of memory required for this page table.

Exercício 24

Calcule o tempo médio para ler uma página de 4096 *bytes* de um disco magnético com as características apresentadas na tabela abaixo.

Seek time médio	Velocidade de de rotação	Taxa de transferência	Taxa de transferência do controlador
10 ms	7500 rpm	90 MB/s	100 MB/s

A quantos ciclos de relógio corresponde o tempo calculado se a frequência do relógio do processador for 1 GHz?

Exercício 25

Seja P um programa cuja execução num único processador demora 20 s, dos quais 1 s corresponde à sua parte sequencial.

a. Qual o *speedup* obtido se o programa for executado em 21 processadores, com o trabalho distribuído igualmente por todos eles?

b. Qual o *speedup* obtido se um dos 21 processadores só fizer 2% do trabalho (paralelizável)?

c. Qual o *speedup* obtido se um dos 21 processadores ficar com 7% do trabalho (paralelizável)?

d. Qual o *speedup* máximo que é possível obter com a paralelização de P?

Exercício 26

Pretende-se calcular o valor de $x_1 \otimes x_2 \otimes \ldots \otimes x_{16}$, onde $\otimes$ é uma operação não paralelizável, cujo cálculo demora tempo t.

a. Considerando que $\otimes$ é uma operação associativa (ie, $(\forall_{a,b,c})\ a \otimes (b \otimes c) = (a \otimes b) \otimes c$), se dispusesse de um número ilimitado de processadores para realizar aquele cálculo, quantos processadores utilizaria, como distribuiria o cálculo por esses processadores, e qual o maior *speedup* que conseguiria obter, em relação ao cálculo sequencial?

b. Qual a relação entre o *speedup* obtido e o *speedup* máximo teórico que seria possível obter com os processadores usados na alínea anterior?

c. Se $\otimes$ não for uma operação associativa (com $a \otimes b \otimes c = (a \otimes b) \otimes c$ e, possivelmente, diferente de $a \otimes (b \otimes c)$), a quantos processadores recorreria para realizar o cálculo no menor tempo possível?

Exercício 6.7 (CODV2)

Consider the following portions of two different programs running at the same time on four processors in a *symmetric multicore processor* (SMP). Assume that before this code is run, both x and y are 0.

```
Core 1: x = 2;
Core 2: y = 2;
Core 3: w = x + y + 1;
Core 4: z = x + y;
```

1. What are all the possible resulting values of w, x, y, and z? For each possible outcome, explain how we might arrive at those values. You will need to examine all possible interleavings of instructions.
2. How could you make the execution more deterministic so that only one set of values is possible?

Exercício 27 (1)

Pretende-se implementar um mecanismo para a sincronização de processos num sistema multiprocessador RISC-V de memória partilhada, que permita garantir que nenhuma *thread* do programa continua a execução, para lá de um ponto de sincronização, até todas as *threads* terem alcançado esse ponto. A base desse mecanismo é uma função `espera` que todas as *threads* devem invocar no ponto de sincronização.

As versões C e RISC-V da implementação proposta para a função são apresentadas abaixo. O valor inicial da variável (global partilhada) `faltam` é o número de *threads* que devem sincronizar naquele ponto.

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

  while (faltam > 0)
    ;
}
```

```
espera: lw   t0, faltam
        addi t0, t0, -1
        sw   t0, faltam
testa:  lw   t0, faltam
        bne  t0, zero, testa
        jalr zero, 0(ra)
```

Exercício 27 (2)

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

  while (faltam > 0)
    ;
}
```

```
espera: lw   t0, faltam
        addi t0, t0, -1
        sw   t0, faltam
testa:  lw   t0, faltam
        bne  t0, zero, testa
        jalr zero, 0(ra)
```

a. Explique a razão por que a implementação proposta pode não funcionar como se pretende.

b. Usando duas *threads* e 2 como valor inicial da variável `faltam`, e tendo como base o código RISC-V acima, mostre uma execução que corresponda ao comportamento descrito na alínea anterior.

c. Apresente uma versão RISC-V da função que tenha o efeito pretendido.

d. Supondo que a execução decorre, enquanto possível, como na alínea b, mostre o que acontece durante a execução do novo código.

Exercício 28

Como poderia programar, em C, a multiplicação de matrizes de inteiros, para ser executada em 4 processadores? Considere que o cabeçalho da função, que calcula `C = A` $\times$ `B`, é o seguinte:

```
void mult(int P, int m, int n, int p,
          int A[m][n], int B[n][p], int C[m][p])
```

onde `P` $\in \{1, 2, 3, 4\}$ é o número do processador onde a função está a ser executada.