Exercícios de preparação para teste

Nas questões de escolha múltipla, existe apenas uma resposta certa.

1. Considere que `W3=0x8ABC0DEF` antes de executar o código seguinte. Indique o valor final de `W3`.

```
str     W3, [X0]
ldrsb   W3, [X0]
```

A. `0x8ABC0DEF` **B. `0xFFFFFFEF`** C. `0x000000EF` D. `00000DEF`

2. Qual das seguintes instruções dá erro de compilação?

A. `SUB X2,X10,X2,ASR #2` B. `ANDS W1,W2,W3` C. `SMULH X0,X0,X0` **D. `LDR W1,[W0,#4]`**

3. Considere o seguinte programa composto por código C e *assembly* `AArch64`.

```
extern unsigned long int SUBR_T2(char *p);

int main(void) {
char s[] = "30 de Abril";
unsigned long int res;

res = SUBR_T2(s);
printf("0x%x\n", res); //Imprime em hexadecimal
return EXIT_SUCCESS;}
```

```
.text
.global SUBR_T2
.type SUBR_T2, "function"

SUBR_T2: LDRB W1, [X0]
         CBZ  W1, FIM
         ADD  X0, X0, #1
         B    SUBR_T2
FIM:     RET
```

Após execução é escrito no ecrã `0x00000000FEFFFFD3`.

Qual o endereço de memória ocupado pelo primeiro elemento ('3') da cadeia de carateres s?

A. `0x00000000FEFFFFD2` **B. `0x00000000FEFFFFC8`**
C. `0x00000000FEFFFFC7` D. `0x00000000FEFFFFBC`

4. Suponha que `x0=0x10000000` e que a tabela representa o conteúdo da memória após executar a instrução `str w9,[x0]`. Qual o conteúdo inicial de `w9` ?

Endereço (hex)	Conteúdo (hex)
10000003	0F
10000002	31
10000001	60
10000000	0C

A. `0x0F31600C`
B. `0xC00613F0`
C. `0xF01306C0`
D. `0x0C60310F`

5. Considere que `SP=0x0805034A0` e que após execução das instruções seguintes o estado da pilha é o indicado na tabela.

```
SMADDL  X1, W1, W1, X1
STR     X1, [SP, #-16]!
```

Endereço (hex.)	Conteúdo (hex.)
0805034A8	0FF
0805034A0	001
080503498	000
080503490	00C

Nestas circunstâncias, pode afirmar-se que o valor inicial de `X1` é:

A. 1 B. -1 C. 0 **D. 3**

6. Considere a declaração `extern int cp(int x);` e a respetiva rotina em *assembly* AArch64:

```
cp:   mov  w2,1
      eor  w3,w3,w3
lpp:  cbz  w0,stop
      and  w4,w0,w2
      add  w3,w3,w4
      lsr  w0,w0,1
      b    lpp
stop: and  w0,w3,w2
      ret
```

Assumindo que a função `cp()` é aplicada a cada elemento da sequência (1,3,5,7) quais os resultados obtidos ?

A. (1,1,1,1) B. (0,0,0,0) C. (1,0,1,0) **D. (1,0,0,1)**

7. Qual das seguintes variantes provoca um erro de compilação?

A. `LDR X9,[X0]` **B. `LDRB X9,[X0]`** C. `LDRSW X9,[X0]` D. `LDRSB X9,[X0]`

8. A instrução `ROR X5, X5, #32` permuta as duas *words* de `X5`. Indique o fragmento de código que <u>não</u> realiza esta operação.

A.
```
UBFX  X6, X5, 32, 32
BFI   X5, X5, 32, 32
ORR   X5, X5, X6
```

B.
```
REV   X5, X5
REV32 X5, X5
```

C.
```
MOV   X6, X5
BFI   X5, X6, 32, 32
UBFX  X5, X6, 32, 32
```

D.
```
MOV   W6, W5
LSR   X5, X5, 32
ADD   X5, X5, X6, LSL 32
```

9. Considere a declaração `extern int ism(int x, int y);` e respetiva rotina em *assembly* AArch64.

```
ism:  sdiv  w2,w0,w1
      msub  w0,w1,w2,w0
      cmp   w0,#0
      cset  w0,eq
      ret
```

Qual o valor de `m = ism(34,7)` ?

A. 1 B. 4 **C. 0** D. 6

10. Considere a declaração `extern int vsum(int *a, int n);` e respetiva rotina em *assembly* AArch64 que pretende calcular a soma dos elementos de um vetor.

```
vsum: eor   x10,x10,x10
nxt:  cbz   x1,stop
      ldrsw x9,[x0],#4
      add   x10,x10,x9
      sub   x1,x1,#1
      b     nxt
stop: mov x0,x10
      ret
```

Qual das afirmações é verdadeira?

A. A rotina está correta.
B. A rotina está errada pois retorna $n \times v[1]$.
C. A rotina está errada pois retorna $n \times v[0]$.
D. A rotina está errada pois retorna $(n-1) \times v[1]$.

11. Considerar a execução do fragmento de código indicado abaixo.

```
L1:   cbz   w1, L2
      ldr   w2, [x6], 4
      sub   w1, w1, 1
      eor   w0, w0, w2
L2:   b     L1
L2:   ...
```

Inicialmente, `w0=0, w1=12`. Quantas instruções são executadas?

R: 61

12. Considere o seguinte programa composto por dois ficheiros.

Ficheiro em linguagem C (`main.c`):

```
extern int SUBROT1(int *a, int d);
int main()
{
  int n = 0, tam = 5;
  int seq[] = {1, 3, 6, 1, 9};
  n = SUBROT1(seq, tam);
  printf("%d\n", n);
  return EXIT_SUCCESS;
}
```

Ficheiro em linguagem *assembly*:

```
SUBROT1:
STP   X29,X30,[SP, #-16]! //<1>
MOV   X29,SP

MOV   W10,0
LDR   W12, [X0]
ADD   X0, X0, #4
SUB   X1, X1, #1
MOV   W11, #1
CICLO:
CBZ   X1, FIM
LDR   W13, [X0]
CMP   W13, W12
B.LE  FSC                    //<2>
ADD   W11, W11, #1
B     PROX
```

```
FSC:
CMP   W10, W11
B.GE  PROX
MOV   W10, W11
MOV   W11, #1

PROX:
MOV   W12, W13
ADD   X0, X0, #4
SUB   X1, X1, #1
B     CICLO

FIM:  MOV   W0, W10
CMP   W10, W11
B.GE  TERMINAR
MOV   W0, W11
TERMINAR:
LDP   X29, X30, [SP], #16
RET
```

(a) Quanta informação, em número de palavras (*words*), é lida de memória ?
A. 9 B. 5 C. 6 D. 7

(b) Assumindo que o valor inicial do registo `SP` é `0xE0`, indique em que endereço da pilha se encontra armazenado o registo `X30` após a execução da instrução assinalada com <1>.
A. 0xE0 B. 0xE8 **C. 0xD8** D. 0xD0

(c) Para os valores de invocação indicados em `main.c`, quantas vezes é tomado o salto assinalado com <2>?
A. 2 **B. 1** C. 3 D. 4

(d) Para os valores indicados no programa, qual é o resultado apresentado no ecrã?
A. 5 B. 2 **C. 3** D. 9

13. Uma memória *cache* com 64 B/bloco contém 32 KiB de dados. Quantos blocos tem?
A. 512 B. 2048 C. 256 D. 1024

14. Qual das seguintes afirmações sobre uma memória *cache* do tipo *write-through* é falsa?
A. O conteúdo da memória principal está sempre atualizado.

B. No caso de uma falta num acesso de leitura (*read miss*) o valor é lido da memória principal e colocado na *cache* atualizando a etiqueta e alterando o valor de **v** para 1.

C. No caso de uma falta num acesso de escrita (*write miss*) o valor é escrito na memória principal e na memória *cache* atualizando a etiqueta e alterando o valor de v para 1.

D. No caso de um acerto num acesso de leitura (*read hit*), o valor é lido da memória *cache*.

15. Um CPU está equipado com uma memória *cache* unificada com taxa de faltas $t_f = 0{,}1$. Em média, 20 % das instruções de um programa acedem a dados. Qual das seguintes alternativas de *split cache* apresenta o mesmo desempenho?

A. I-*cache* com $t_f = 20\,\%$ e D-*cache* com $t_f = 8\,\%$;

B. I-*cache* com $t_f = 8\,\%$ e D-*cache* com $t_f = 20\,\%$;

C. I-*cache* com $t_f = 5\,\%$ e D-*cache* com $t_f = 5\,\%$;

D. I-*cache* com $t_f = 10\,\%$ e D-*cache* com $t_f = 20\,\%$.

16. A memória principal usada com um CPU de 2 GHz tem um tempo de acesso de 60 ns. O acesso à memória *cache* demora 0,5 ns. Qual deve ser o valor máximo da taxa de faltas para que o tempo médio de acesso a memória seja 10 ciclos?

A. 5 % **B. 7,5 %** C. 10 % D. 2,5 %

17. Qual das seguintes afirmações sobre uma memória *cache* do tipo *write-back* é verdadeira?

A. Existe a possibilidade de serem feitos 2 acessos a memória principal para apenas um acesso a memória *cache*.

B. O conteúdo da memória principal está sempre atualizado.

C. Pode haver um acesso a memória principal mesmo que o acesso a memória *cache* seja um acerto (*hit*).

D. Não pode ser usada como *D-cache*.

18. O desempenho de uma memória *cache* unificada usada com um CPU de 1 GHz foi considerado insuficiente. Sabendo que o CPI de base é 1, indique a alteração que leva à maior redução do CPI efetivo.

A. Baixar a taxa de instruções *load/store* de 50 % para 20 %.

B. Baixar o tempo de acesso à memória externa de 100 ns para 80 ns.

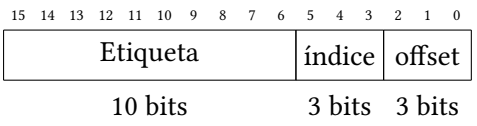
C. Baixar a taxa de faltas de 15 % para 10 %.

D. Nenhuma das outras alterações indicadas reduz o CPI efetivo.

19. A tabela seguinte apresenta o conteúdo (em hexadecimal) de uma memória *cache* do tipo *write-back* com 8 blocos de 8 bytes usada como *D-cache* num CPU com endereços de 16 bits.

bloco	conteúdo 7 6 5 4 3 2 1 0	etiqueta	v	d
0	aa cc de hf 34 33 11 01	235	1	0
1	bb ad 45 4f af de 21 99	391	1	1
2	cc 34 ab 1f 56 cd ff ff	023	1	1
3	dd 67 22 2b 32 56 32 21	198	0	1
4	ee 32 11 9f aa ba ab bb	311	1	0
5	ff 10 00 04 01 02 03 04	278	0	0
6	11 03 41 32 cc dd ee ff	212	1	1
7	22 01 65 01 05 06 07 08	387	0	1

(a) Como é decomposto o endereço para acesso à memória *cache*? Justifique.



Offset: 3 bits. Como o conteúdo de cada bloco tem 8 bytes, serão necessários 3 bits para designar um byte do bloco.
Índice: 3 bits. Como a cache possui 8 blocos, são necessários 3 bits para representar os índices de 0 a 7.
Etiqueta: 10 bits. Como temos 16 bits e já gastámos 6, os restantes ($16 - 6 = 10$) serão para a etiqueta.

(b) Indique (se possível) o valor (byte) em memória principal no endereço `0xc467`. Justifique.

Decompondo o endereço temos: `0xC467` → `1100 0100 0110 0111` → `1100010001 100 111`.
Portanto, etiqueta: 311_{16}, índice: 4, *offset*: 7.

As etiquetas são iguais e o bloco é válido porque `v=1` (*read hit*).

O valor em memória *cache* é `0xEE` (elemento 7 do bloco 4) e corresponde ao valor em memória principal porque `d=0`.

(c) Explique quais as alterações que ocorrem na *cache* e na memória principal durante a leitura do valor (byte) residente no endereço `0xe48d`.

Decompondo o endereço temos: `0xE48D` → `1110 0100 1000 1101` → `1110010010 001 101`.
Portanto, etiqueta: 392_{16}, índice: 1, *offset*: 5.

Como as etiquetas são diferentes, ocorre uma falta de leitura *read miss*. Os dados do bloco são válidos (`v=1`), mas não correspondem à posição de memória pretendida.

Como `v=1` e `d=1` é necessário fazer *write-back*: o valor `0x45` é escrito na memória principal no endereço `0xE44D` (endereço calculado com a etiqueta residente em *cache*).

É necessário ler de memória principal o bloco que contém o valor no endereço `0xE48D`, escrevendo-o na posição 1 da *cache* (posição 5) e atualizando a etiqueta desse bloco para `392`. Os indicadores devem ficar com os valores `v=1` e `d=0`.

20. Um CPU com endereços de 24 bits está equipado com uma memória *cache* de dados do tipo *write-through*. Etiqueta e índice têm, respetivamente, 14 e 6 bits de comprimento.

(a) Determinar o número de blocos e o número de bytes por bloco desta memória *cache*.

Como o índice tem 6 bits, a memória *cache* tem 2^6=64 blocos.
Como ficam 24-14-6=4 bits para o deslocamento, cada bloco tem 2^4=16 bytes por bloco.

(b) Considerar a seguinte situação. São realizadas sucessivamente leituras (de uma palavra) das seguintes posições de memória:

`0x3B7C94`, `0x3B6C90`, `0x3B6C98`, `0x3B6C94`

Quantos blocos são transferidos de memória principal para memória *cache* por causa dos três últimos acessos?

A primeira leitura (endereço `0x3B7C94`) afeta o bloco 9 (001001_2):

0 1 2 3 4 5 6 7 8 9 10 11 12 13	14 15 16 17 18 19	20 21 22 23
00111011011111	001001	1000
etiqueta	índice	desl.

Seja *hit* ou *miss*, a etiqueta do bloco 9 é 00111011011111_2 após a execução do acesso.
A segunda leitura (endereço `0x3B6C90`) afeta o mesmo bloco (9):

00111011011011	001001	0000

Como a etiqueta é diferente daquela que está em *cache*, trata-se de uma falta (*miss*) e é lido um bloco de memória principal. A etiqueta em memória *cache* é, agora, 00111011011011_2.
Os dois endereços seguintes levam sempre a *hit*, já que correspondem a partes do mesmo bloco (etiqueta e índices iguais aos do 2º acesso).

00111011011011	001001	1000
00111011011011	001001	0100

Logo, não existem mais acessos a memória principal: os últimos três acessos provocam a leitura de 1 bloco da memória principal.

(c) A memória *cache* é usada num sistema em que a penalidade de faltas é de 80 ciclos. Qual deve ser o valor máximo da taxa de faltas desta *cache* para que o número médio de ciclos de protelamento *no acesso a dados* não exceda 10?

O número de ciclos de protelamento (por operação de acesso a dados) C_p é dado pela número médio de acessos a dados que resultam em falta vezes a penalidade p_f (a dividir pelo número de acessos a dados N_d.

$$C_p = \frac{(N_d \times m_d) \times p_f}{N_d} = m_d \times p_f$$

Das condições do enunciado tem-se:

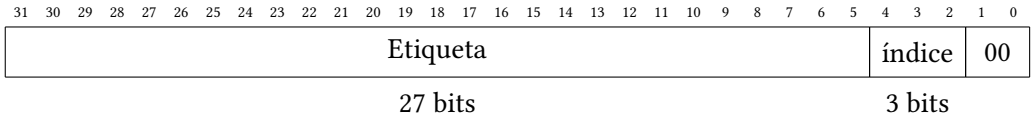
$$m_d \times 80 \leq 10 \qquad \Rightarrow \qquad m_d \leq= \frac{10}{80} = 0{,}125$$

Portanto, $m_d \leq 12{,}5\,\%$.

21. A tabela seguinte apresenta o conteúdo de uma memória *cache* do tipo *write-through* com 8 blocos de 4 bytes usada como *D-cache* num CPU com endereços de 32 bits.

	Conteúdo	Etiqueta	v
0	aa ff cc 33	123456a	0
1	12 34 56 78	7bcd001	1
2	88 b0 3c 2b	7fffd55	1
3	71 ab 3f 6d	7fffd55	1
4	34 ff 13 aa	07f9910	0
5	78 00 9c 23	0000893	1
6	7a 10 9f a3	2900002	1
7	99 43 65 b4	7f01d12	0

(a) Como é decomposto o endereço para acesso à memória *cache*? Justifique.



Offset: 2 bits.

Índice: 3 bits. Como a cache possui 8 blocos são necessários 3 bits para representar os índices de 0 a 7.

Etiqueta: 27 bits. Como temos 32 bits e já gastamos 5, os restantes ($32 - 5 = 27$) serão para a etiqueta.

(b) Apresente, justificando, o conteúdo da memória *cache* após a execução das seguintes operações (se em algum caso não for possível conhecer o conteúdo escreva "indeterminado"). Cada registo `Rx` tem 32 bits.

```
1: R1 ← 0xfafafafa
2: R4 ← R3+R2
3: R4 ← 0x0ff32210
4: R5 ← MEM[R4]
5: R5 ← R4-R1
6: R6 ← 0xffffaaa8
7: MEM[R6+4] ← R1
```

	Conteúdo	Etiqueta	v
0	aa ff cc 33	123456a	0
1	12 34 56 78	7bcd001	1
2	88 b0 3c 2b	7fffd55	1
3	fa fa fa fa	7fffd55	1
4	indeterminado	07f9910	1
5	78 00 9c 23	0000893	1
6	7a 10 9f a3	2900002	1
7	99 43 65 b4	7f01d12	0

As únicas instruções que acedem à memória de dados são:

- Instrução 4:
  Endereço `0x0ff32210 -> 00001111111100110010001000 | 100 | 00`
  Etiqueta: `0x07f9910`, bloco 4.
  Apesar de a etiqueta ser igual, tem-se $v = 0$ *(read miss)*. É necessário ler o valor da memória principal e escrevê-lo na memória cache, colocando ainda $v = 1$.
- Instrução 7:
  1º passo: Calcular o endereço: `0xffffaaa8+4=0xffffaaac`.
  2º passo: Obter etiqueta e bloco: `0xffffaaac` -> 11111111111111110101010101 | 011 | 00
  Etiqueta `0x7fffd55`, bloco 3.
  3º passo: Verificar se é para alterar a memória *cache*.
  O bloco 3 tem etiqueta igual e $v = 1$ *(write hit)*. O valor de `t1` é escrito na memória principal e também na memória *cache*: o conteúdo do bloco 3 é alterado para `0xfafafafa`.

22. Um CPU tem endereços de 20 bits e usa uma memória *cache* com 1 palavra/bloco. A memória *cache* do tipo *write-back* usa 6 bits para cada índice e 12 bits para cada etiqueta. Uma parte do conteúdo da memória *cache* está indicada (em hexadecimal) na tabela seguinte:

	conteúdo	etiqueta	v	d
0	12345678	ABC	1	1
1	6548FEAB	123	0	0
2	3C1F56FD	678	1	0
3	AFD12498	567	1	1
4	6198FA34	B7C	1	0
5	1929AAAA	8D1	0	1
...	...	...	...	...

(a) Determinar o número de blocos e o número total de bits usados na memória *cache*.

> Como o número de bits do índice é 6, o número de blocos é $2^6 = 64$.
> Cada bloco tem 32 bits de dados, 12 bits de etiqueta, 1 bit de validade e 1 bit de atualização (d), o que dá um total de 46 bits por bloco.
> Logo, o número total de bits é $64 \times 46 = 2944$.

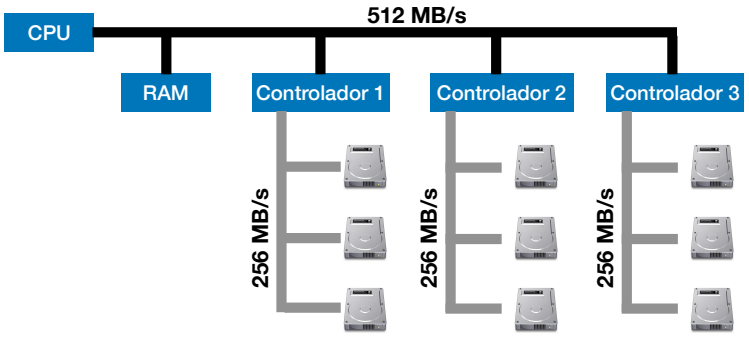
(b) Determinar, se possível, a posição em memória do valor `0x6198FA34`.

> O valor está no bloco 4 da memória *cache* com etiqueta *0xb7c*. Este valor é válido $(v = 1)$ e igual ao valor em memória principal $(d = 0)$.
> Portanto, o valor `0x6198fa34` está na posição `1011 0111 1100 | 0001 00 | 00`, i.e., `0xB7C10`.

(c) Determinar, se possível, qual o valor atualmente armazenado na posição de memória `0x5670C`.

> O endereço em binário é `0101 0110 0111 | 0000 11 |00`. Logo, se a memória *cache* contiver o conteúdo desejado será no bloco 3.
> O bloco 3 é válido porque $v = 1$ e a sua etiqueta `0x567` é idêntica à do endereço indicado.
> Contudo, como $d = 1$, o seu conteúdo é (potencialmente) diferente do valor atual ainda guardado em memória.
> Logo, não é possível determinar o valor ainda armazenado na memória principal (o valor em memória *cache* é mais recente)..

23. Considere o computador indicado na figura e que tem as seguintes caraterísticas:



- O CPU opera a 2 GHz;
- O barramento de memória possui uma taxa de transferência de 512 MB/s;
- Ligados ao barramento de memória estão 3 controladores de barramento SCSI Ultra32 com uma taxa de tranferência de 256 MB/s, cada um com 3 discos;
- O acesso aos discos é feito com uma largura de banda de 55 MB/s e o tempo médio de busca mais a latência de rotação é 6 ms;

- O acesso aos discos é feito em blocos de 512 kB, guardados em setores consecutivos;
- Em cada acesso, o programa do utilizador e o sistema operativo gastam, respetivamente, 1 milhão e 1,5 milhões de ciclos de relógio.

Determine qual dos recursos (CPU, barramento de memória ou discos) limita o desempenho expresso em blocos processados por unidade de tempo. [Considere kB = 10^3 B, MB = 10^6 B.]

CPU:

Tratamento de 1 bloco:

$$\frac{1 \times 10^6 + 1,5 \times 10^6}{2 \times 10^9} = 1{,}25\,\text{ms}$$

Por segundo: 800 blocos

Barramento de Memória:

$$\frac{512\,\text{MB/s}}{512\,\text{kB}} = 1000\,\text{blocos/s}$$

Discos:

Por disco:

$$6\,\text{ms} + \frac{512\,\text{kB}}{55\,\text{MB/s}} = 15{,}3\,\text{ms}$$

O que corresponde a 65 blocos por disco por segundo, no total (por controlador) temos então:

$$3 \times 65\,\text{blocos/s} = 195\,\text{blocos/s}$$

Um a vez que temos 3 controladores temos no total:

$$3 \times 195\,\text{blocos/s} = 585\,\text{blocos/s}$$

Como o barramento do controlador tem uma taxa de tranferência de 256 MB/s, temos de verificar se ela é suficiente:

$$\frac{256\,\text{MB/s}}{512\,\text{kB}} = 500\,\text{blocos/s}$$

Uma vez que os 3 discos só transferem 195 blocos/s a largura de banda do barramento do controlador é suficiente.

Como os discos transferem 585blocos/s, o barramento de memória suporta a transferência de 1000 blocos/s e o CPU 800 blocos/s são os discos que limitam o desempenho.

24. Um computador possui um CPU que opera a 2,5 GHz e está equipado com um disco que transfere grupos de 8 palavras (4 B cada) a uma taxa de 16 MB/s. [Considere kB = 10^3 B, MB = 10^6 B.]
    (a) O acesso ao disco é feito pela técnica de *polling*. Determinar a fração de tempo de CPU consumida, assumindo que cada operação de *polling* gasta 1000 ciclos de relógio.

Dados transferidos por acesso:

$$8 \times 4\text{B} = 32\,\text{B}$$

Acessos por segundo:

$$\frac{16\,\text{MB/s}}{32\,\text{B}} = \frac{16 \times 10^6}{2 \times 16} = 0{,}5 \times 10^6$$

Então, o número de ciclos consumidos pela operação é de:

$$0,5 \times 10^6 \ \times 1000 = 5 \times 10^8$$

Tendo em consideração o número de ciclos consumidos por segundo, a percentagem média de tempo de CPU gasto é:

$$\frac{5 \times 10^8}{2{,}5 \times 10^9} = 0{,}2 = 20\,\%$$

(b) O disco do computador só transfere dados em 20% do tempo. Calcular a fração de tempo de CPU consumida recorrendo à técnica de interrupção. Assumir que o *overhead* de cada transferência, incluindo o atendimento da interrupção, é de 2000 ciclos de relógio.

Dados transferidos por acesso:

$$8 \times 4\text{B} = 32\,\text{B}$$

Acessos por segundo:

$$0{,}2 \times \frac{16\,\text{MB/s}}{32\,\text{B}} = 0{,}2 \times \frac{16 \times 10^6}{2 \times 16} = 0{,}2 \times 0{,}5 \times 10^6 = 10^5$$

Então, o número de ciclos consumidos pela operação é de:

$$10^5 \times 2000 = 2 \times 10^8$$

Tento em consideração o número de ciclos consumidos por segundo, a percentagem média de tempo de CPU gasto é:

$$\frac{2 \times 10^8}{2{,}5 \times 10^9} = 0{,}08 = 8\,\%$$

(c) Tendo em consideração os resultados das alíneas anteriores, determinar a partir de que percentagem de ocupação do disco é mais vantajoso o uso da técnica de *polling*.

Sabendo que a técnica de *polling* consome 20% do CPU, então temos de terminar a partir de que percentagem de ocupação do disco a técnica de interrupções consome mais de 20%.

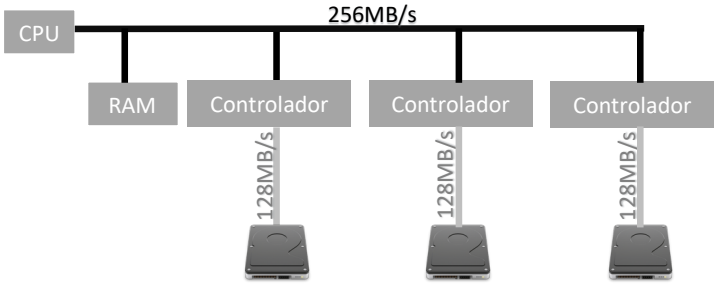
$$\frac{X \times 0{,}5 \times 10^6 \times 2000}{2{,}5 \times 10^9} > 0{,}2$$

$$X > 0{,}5$$

Logo, a técnica de *polling* é mais vatajosa se o disco disco estiver ocupado em mais de 50% do tempo.

question[30]

Considere o computador indicado na figura e que tem as seguintes caraterísticas:



- O CPU opera a 3 GHz;
- O barramento de memória possui uma taxa de transferência de 256 MB/s;
- Ligados ao barramento de memória estão 3 controladores, cada um com o seu disco;
- O acesso aos discos é feito com uma largura de banda de 128 MB/s e o tempo médio de busca mais a latência de rotação é 3 ms;
- O acesso aos discos é feito em blocos de 256 kB, guardados em setores consecutivos;
- Em cada acesso, o programa do utilizador e o sistema operativo gastam, respetivamente, 1 milhão e 2 milhões de ciclos de relógio.

Determine qual dos recursos (CPU, barramento de memória ou discos) limita o desempenho expresso em blocos processados por unidade de tempo. [Considere kB = 10^3 B, MB = 10^6 B.]

CPU:

Tratamento de 1 bloco:

$$\frac{1 \times 10^6 + 2 \times 10^6}{3 \times 10^9} = 1\,\text{ms}$$

Por segundo: 1000 blocos

Barramento de Memória:

$$\frac{256\,\text{MB/s}}{256\,\text{kB}} = 1000\,\text{blocos/s}$$

Discos:

Por disco:

$$3\,\text{ms} + \frac{256\,\text{kB}}{128\,\text{MB/s}} = 5\,\text{ms}$$

O que corresponde a 200 blocos por disco por segundo, no total temos então:

$$3 \times 200\,\text{blocos/s} = 600\,\text{blocos/s}$$

Como o CPU consegue processar 1000 blocos/s e o barramento de memória suporta também a transferência de 1000 blocos/s, são os discos que limitam o desempenho.

25. Um sistema possui um CPU que opera a 4 GHz. Este sistema possui ainda um disco que transfere dados para o processador em grupos de 4 palavras (2 bytes cada) e tem uma taxa de transferência de dados de 40 MB/s. [Considere kB = 10^3 B, MB = 10^6 B.]

(a) Se pretendermos utilizar *polling* como técnica de gestão de periféricos e sabendo que uma operação de *polling* consome 400 ciclos de relógio, qual é a fração de tempo de CPU consumida?

Dados transferidos por acesso:

$$4 \times 2\text{B} = 8\,\text{B}$$

Acessos por segundo:

$$\frac{40\,\text{MB/s}}{8\,\text{B}} = \frac{40 \times 10^6}{8} = 5 \times 10^6$$

Então, o número de ciclos consumidos pela operação é de:

$$5 \times 10^6 \ \times 400 = 2 \times 10^9$$

Tento em consideração o número de ciclos consumidos por segundo, a percentagem média de tempo de CPU gasto é:

$$\frac{2 \times 10^9}{4 \times 10^9} = 0{,}5 = 50\,\%$$

(b) Se em vez de *polling* fosse utilizada a técnica de interrupções, qual seria a fração de tempo de CPU consumida? Admita que o *overhead* de cada transferência, incluindo o atendimento da interrupção, é de 600 ciclos de relógio e que o disco está sempre potencialmente ocupado.

Dados transferidos por acesso:

$$4 \times 2\text{B} = 8\,\text{B}$$

Acessos por segundo:

$$\frac{40\,\text{MB/s}}{8\,\text{B}} = \frac{40 \times 10^6}{8} = 5 \times 10^6$$

Então, o número de ciclos consumidos pela operação é de:

$$5 \times 10^6 \ \times 6 \times 10^2 = 3 \times 10^9$$

Tento em consideração o número de ciclos consumidos por segundo, a percentagem média de tempo de CPU gasto é:

$$\frac{3 \times 10^9}{4 \times 10^9} = 0{,}75 = 75\,\%$$

(c) Comparando os resultados das alíneas anteriores, em que situações é que a técnica de interrupções pode ser vantajosa?

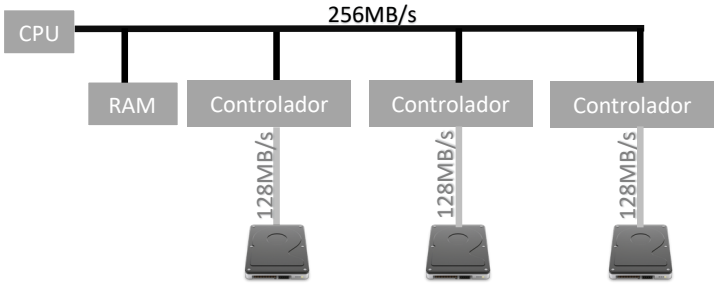
No segundo cenário, o custo (em ciclos de relógio) do *overhead* de cada transferência, incluindo o atendimento da interrupção, é superior ao custo de uma operação de *polling*. A técnica de interrupções é mais vantajosa quando o periférico não está sempre potencialmente ocupado. Para este caso concreto, a técnica de interrupções é mais vantajosa quando o disco está ocupado menos de 66,7 % do tempo.

$$\frac{X \times 3 \times 10^9}{4 \times 10^9} = 0{,}5$$

$$X \times \frac{3}{4} = 0{,}5$$

$$X = \frac{0{,}5}{0{,}75} = 66\,\%$$

26. Considere o computador indicado na figura e que tem as seguintes caraterísticas:



- O CPU opera a 3 GHz;
- O barramento de memória possui uma taxa de transferência de 256 MB/s;
- Ligados ao barramento de memória estão 3 controladores, cada um com o seu disco;
- O acesso aos discos é feito com uma largura de banda de 128 MB/s e o tempo médio de busca mais a latência de rotação é 3 ms;
- O acesso aos discos é feito em blocos de 256 kB, guardados em setores consecutivos;
- Em cada acesso, o programa do utilizador e o sistema operativo gastam, respetivamente, 1 milhão e 2 milhões de ciclos de relógio.

Determine qual dos recursos (CPU, barramento de memória ou discos) limita o desempenho expresso em blocos processados por unidade de tempo. [Considere kB = 10^3 B, MB = 10^6 B.]

CPU:

Tratamento de 1 bloco:

$$\frac{1 \times 10^6 + 2 \times 10^6}{3 \times 10^9} = 1\,\text{ms}$$

Por segundo: 1000 blocos

Barramento de Memória:

$$\frac{256\,\text{MB/s}}{256\,\text{kB}} = 1000\,\text{blocos/s}$$

Discos:

Por disco:

$$3\,\text{ms} + \frac{256\,\text{kB}}{128\,\text{MB/s}} = 5\,\text{ms}$$

O que corresponde a 200 blocos por disco por segundo, no total temos então:

$$3 \times 200\,\text{blocos/s} = 600\,\text{blocos/s}$$

Como o CPU consegue processar 1000 blocos/s e o barramento de memória suporta também a transferência de 1000 blocos/s, são os discos que limitam o desempenho.

27. O computador de bordo de uma boia oceanográfica envia por rádio, a cada n minutos, a posição da boia e um conjunto de informações meteorológicas relevantes, num total de 2500 Bytes. O rádio tem uma potência de 180 W e envia informação a uma cadência de 10 kbit/s; o restante `hardware` tem uma potência de 1 W. A bateria tem capacidade para armazenar 1200 W h de energia. A boia é lançada ao mar com a bateria totalmente carregada.

    (a) Determine qual deve ser a periodicidade do envio de informação (valor de n) por forma a que a boia possa navegar durante 25 dias. Note que o número de envios por hora é dado por $60/n$.

    Tempo de envio de 2500 Bytes: $\frac{2500 \times 8}{10 \times 10^3} = \frac{20 \times 10^3}{10 \times 10^3} = 2\text{s} = \frac{2}{3600}\text{h}$

    Energia por envio: $180 \times \frac{2}{3600} = 0{,}1\,\text{W h}$; envios por hora: $\frac{60}{n}$

    Energia por hora: $1 + 0{,}1 \times \frac{60}{n} = \left(1 + \frac{6}{n}\right)\text{W h}$

    Em 25 dias: $25 \times 24 \times \left(1 + \frac{6}{n}\right) = 1200 \Leftrightarrow 1 + \frac{6}{n} = 2 \Leftrightarrow n = 6$

    A periodicidade será pois um envio a cada 6 minutos, isto é, 10 envios por hora.

    (b) A cadência de envio da informação é agora de minuto a minuto e a boia foi equipada com painéis fotovoltaicos capazes de fornecer, em média, 160 W h de energia por dia. Determine quantos dias a boia se manterá em funcionamento.

    Energia por envio: $180 \times \frac{2}{3600} = 0{,}1\,\text{W h}$; envios por hora: 60

    Energia por hora: $1 + 0{,}1 \times 60 = 7\text{W h}$

    Em 24h, gasta: $24 \times 7 = 168\,\text{W h}$; reposta: 160W h; saldo: 8W h gastos por dia

    Duração da bateria: $\frac{1200}{8} = 150$ dias

    A boia poderá navegar durante 150 dias.

28. Um sistema composto por um CPU, que opera a 1 GHz e um disco duro que transfere dados em grupos de 4 palavras (8 bytes cada) a uma taxa de 8 MB/s, utiliza o método de comunicação com periféricos conhecido como interrupções. Assumindo que o *overhead* de cada transferência, incluindo o atendimento da interrupção, é de 2000 ciclos de relógio e que o disco duro transfere dados durante 10 % do tempo, calcule a percentagem de tempo médio de CPU consumido nas transferências. [Considere kB = 10^3 B, MB = 10^6 B.]

    Dados transferidos por acesso:

    $$4 \times 8\text{B} = 32\,\text{B}$$

    Acessos por segundo:

    $$\frac{8\,\text{MB/s}}{32\text{B}} = \frac{1 \times 10^6}{4 \times 8} = 0{,}25 \times 10^6$$

Uma vez que o disco só transfere dados em 10% do tempo o numero de acessos será de:

$$0{,}10 \times 0{,}25 \times 10^6 = 25 \times 10^3$$

Então o número de ciclos consumidos pela operação será de:

$$25 \times 10^3 \ \times 2 \times 10^3 = 5 \times 10^7$$

Tento em conta o número de ciclos consumidos por segundo, a percentagem média de tempo de CPU consumida pela técnica será de:

$$\frac{5 \times 10^7}{1 \times 10^9} = 5 \times 10^{-2} = 0{,}05 = 5\,\%$$