Este exame tem 6 questões, num total de 200 pontos. Responda em folhas separadas a cada um dos seguintes conjuntos de problemas: (1), (2 e 3), (4), (5 e 6).

1. O seguinte programa em linguagem C invoca uma sub-rotina em linguagem *assembly* AArch64.

   Ficheiro *main.c*

```c
#include <stdio.h>
extern unsigned long collatz(unsigned long n);
int main (void)
{
   printf("Resultado: %ld\n",collatz(10));
   return 0;
}
```

   Ficheiro *collatz.s*

```
1  .text
2  .global collatz
3  .type collatz, %function
4
5  collatz:
6      mov     x1, #0
7  L1: cmp     X0, #1
8      b.eq    L3
9      add     X1, X1, #1
10     ands    X2, X0, #1
11     b.eq    L2
12     mov     X4, X0
13     add     X0, X0, x0
14     add     X0, X0, X4
15     add     X0, X0, 1
16     b       L1
17 L2: lsr     X0, X0, 1
18     b       L1
19 L3: mov     X0, X1
20     ret
```

   Justificar todas as respostas.

[10] (a) O bloco de código das linhas 12–15 modifica o valor do registo `X0`. Determinar a relação entre o valor inicial e final do registo.

> **Resposta:** O valor inicial de `X0` é guardado em `X4`. De seguida, o valor de `X0` é duplicado por adição de `X0` consigo próprio. A esse valor é adicionado o valor original de `X0` (vindo de `X4`).

Portanto, `X0` contém agora o triplo do valor inicial. A última instrução acrescenta uma unidade a este valor.

$$\text{Valor\_final} = 3 \times \text{Valor\_inicial} + 1$$

[10] (b) No contexto do programa, qual é o objetivo das linhas 10–11?

Resposta: Como o valor 1 apenas tem o bit menos significativo a 1, a operação `AND` garante que apenas o bit menos significativo de `X1` pode ser diferente de zero. Se for zero (`X2=0`), i.e., se o valor de `X1` for par, é feito um salto para `L2`.

Portanto, estas linhas determinam se número em `X0` é par ou não.

[10] (c) Quantas vezes é executada a instrução com a etiqueta `L2`?

Resposta: Essa instrução está contida num ciclo (com início em `L1`) e é sempre executada quando o valor em `X0` é par. Essa instrução divide o valor de `X0` por dois. Caso `X0` seja ímpar, é calculado um novo valor conforme descrito na alínea (a). O ciclo termina quando `X0` atingir o valor 1.

Para o valor inicial `X0=10`, os valores deste registo são sucessivamente:

$$10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

A divisão por 2 é realizada 5 vezes.

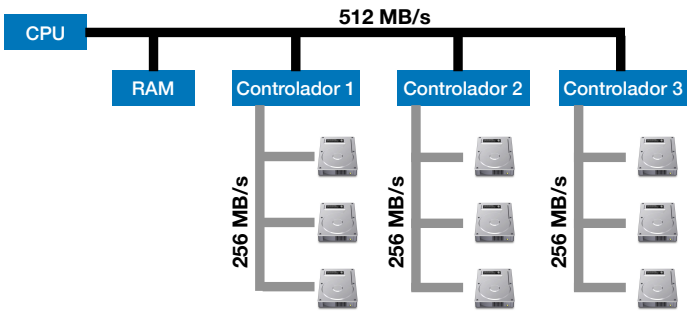
[10] (d) Que mensagem é apresentada no monitor antes do programa terminar?

Resposta: O resultado da sub-rotina é o valor acumulado em `X1` (e transferido para `X0` no final). `X1` começa a 0 e é incrementado uma vez por iteração (é um contador). Ou seja, o resultado de `collatz` é o número de iterações necessário para `X0` atingir o valor 1.

Conforme indicado na solução da alínea (c), são necessárias 6 iterações. Logo, a mensagem apresentada é:

```
Resultado: 6
```

[30] 2. 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.]

Resposta:

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.

[30] 3. Considerar a função $f(x), x \in \mathbb{R}$, definida por

$$f(x) = \begin{cases} -2x + \sqrt{x} & \text{se} \quad x \geq 2 \\ x^3 & \text{se} \quad x < 2 \end{cases}$$

Implementar a sub-rotina `Func` que calcula o valor da função para qualquer valor de x. Considerar que o protótipo da função a invocar em C é: `double Func(double x)`. **Atenção:** Deverá implementar a sub-rotina sem recorrer à declaração de constantes.

Resposta:

```
Func:
     MOV X1, 2
     SCVTF D1, X1
     FCMP D0, D1
     BLT RAMO2
     FNMUL D1, D1, D0
     FSQRT D0, D0
     FADD D0, D0, D1
     B FIM
RAMO2:
     FMUL D1, D0, D0
     FMUL D0, D0, D1
FIM:
 RET
```

4. Um conjunto não vazio de n pontos (x_i, y_i), no plano é representado por uma sequência de números inteiros organizados pela seguinte ordem: $(x_1, y_1, x_2, y_2, x_3, y_3, \ldots, x_n, y_n)$.

   Esta sequência de `n` pontos forma um vetor em memória endereçado por `P`.

[20] (a) Escrever a sub-rotina `menorX` que determina a menor coordenada x_i de um conjunto de pontos representado conforme indicado. Esta sub-rotina seria invocada de C segundo o protótipo:

```
long int menorX(long int *P, int n);
```

Resposta:

```
menorX: LDR  X2, [X0], #16
L1x:    SUB  W1, W1, #1
```

```
        CBZ  W1, L2x
        LDR  X3, [X0], #16
        CMP  X2, X3
        BLT  L1x
        MOV  X2, X3      // Atualiza mínimo
        B    L1x
L2x:    MOV  X0, X2
        RET
```

[30] (b) Assumir que existe a sub-rotina `maiorY` que determina a maior coordenada y_i de um conjunto de pontos. O protótipo da sub-rotina em C é: `long int maiorY(long int *P, int n)`.

Implementar a sub-rotina `PONTO` que, utilizando as sub-rotinas `menorX` e `maiorY`, determina se algum ponto com a menor coordenada x_i do conjunto também tem a maior coordenada y_i. Em caso afirmativo a sub-rotina devolve 1, caso contrário devolve 0.

Esta sub-rotina pode ser invocada de C segundo o protótipo:

```
int PONTO(long int *P, int n);
```

Resposta:

```
PONTO: STP  X29, X30, [SP, #-64]! // Guarda FP e LR e reserva espaço para outros
       STP  X0, X1, [SP, #16]     // Guarda P e n
       STP  X20, X21, [SP, #32]   // Preserva X20 e X21
       STP  X22, X23, [SP, #48]   // Preserva X22 e X23

       BL   menorX
       MOV  X20, X0               // Resultado de menorX em X20
       LDP  X0, X1, [SP, #16]     // Restabelece P e n
       BL   maiorY
       MOV  X21, X0               // Resultado de maiorY em X21
       LDP  X0, X1, [SP, #16]     // Restabelece P e n
L1:    CBZ  W1, L2
       LDP  X22, X23, [X0], #16   // Lê coordenadas de ponto
       CMP  X22, X20              // xi = menor x?
       BNE  L3
       CMP  X23, X21              // yi = maior y?
       BNE  L3
       MOV  X0, #1                // Sai com X0=1
       B    L4
L3:    SUB  X1, X1, #1
       B    L1
L2:    MOV  X0, #0                // Sai com X0=0
L4:    LDP  X20, X21, [SP, #32]   // Repõe X20 e X21
```

```
        LDP  X22, X23, [SP, #48]   // Repõe X22 e X23
        LDP  X29, X30, [SP], #64
        RET
```

[20] 5. Implementar em *assembly* Aarch64 (NEON), a sub-rotina

`unsigned int words(unsigned char *txt, unsigned int n)`

que conta o número de palavras contidas num texto `txt` de dimensão `n`. Assumir que o texto não começa nem termina com espaços e que só há um espaço entre palavras. Para simplificar assuma que n é múltiplo de 16.

Resposta:

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

words:
    lsr w1, w1, #4          // iterações = n/16
    mov w2, #' '            // código ASCII do espaço
    dup v1.16b, w2          // replica-o 16 vezes no vetor v1
ciclo:
    cbz x1, fim
    ldr q0, [x0], #16   // empacota 16 letras e aponta para as próximas 16
    cmeq v2.16b,v1.16b,v0.16b   // compara com 16 espaços; resultado em v2
    addv b3, v2.16b         // contabiliza espaços existentes
    smov w3, v3.b[0]        // w3 é o simétrico do número de espaços
    sub x4, x4, x3          // acumula em x4
    sub x1, x1, #1
    b ciclo
 fim:
    add x0, x4, #1
    ret
```

Questões de escolha múltipla

6. Cada uma das seguintes questões tem apenas uma resposta certa. Indique as respostas corretas **na folha de resposta**.

[5] (a) Sobre transferências por acesso direto à memória (DMA), qual das afirmações é verdadeira?

A. O CPU configura a transferência e é avisado quando esta acaba.

B. O CPU não participa na transferência.

C. O CPU efetua a transferência.

D. O CPU configura a transferência mas não é avisado quando esta acaba.

[5] (b) Considere a função `extern int vsum(double *a, float *b, int n, float c)` implementada em *assembly* `AArch64`. Qual a correspondência correta entre parâmetros e registos?

A. `X0,X1,W2,S0` B. `D0,S0,W0,S1` C. `X0,X1,W2,S3` D. `D0,S1,W0,S2`

[5] (c) Indique qual o maior obstáculo ao aproveitamento efetivo de multi-processadores de memória partilhada?

A. o tempo de acesso à memória **B. a lei de Amdahl**
C. a utilização de instruções SIMD D. o consumo de energia

[5] (d) Considerar um sistema para computação científica cujo desempenho de pico de 16 GFLOPs/s é atingido para uma intensidade aritmética de 4 FLOPs/byte se nenhuma otimização de software for usada.

Qual das seguintes otimizações a implementar pelo compilador pode melhorar o desempenho de uma tarefa de intensidade aritmética de 2 FLOPs/byte?

A. Melhor aproveitamento do paralelismo ao nível das instruções
B. Maior utilização de instruções SIMD
C. Melhor aproveitamento da afinidade de memória
D. Nenhuma das outras opções

[5] (e) Quantos bytes são reservados pela declaração:

```
.hword 31, -5, 87, 90, -3, -90
```

A. 6 **B. 12** C. 18 D. 24

[5] (f) Os multi-processadores de memória partilhada são computadores do tipo:

A. SISD B. SIMD **C. MIMD** D. MISD