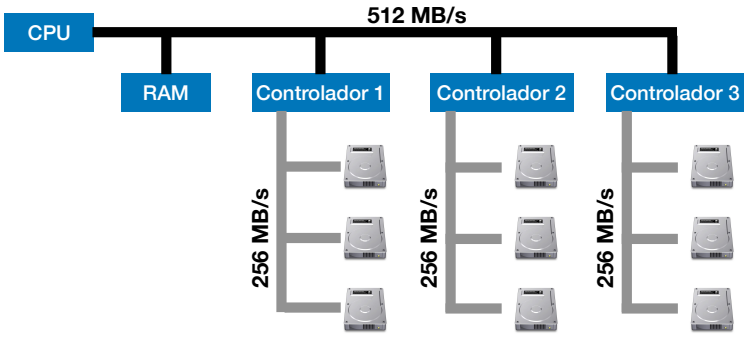
Questões-tipo para o exame de MPCP

As questões apresentadas a seguir são exemplos de questões que poderiam estar no exame de MPCP.

1. 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.

2. 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%.

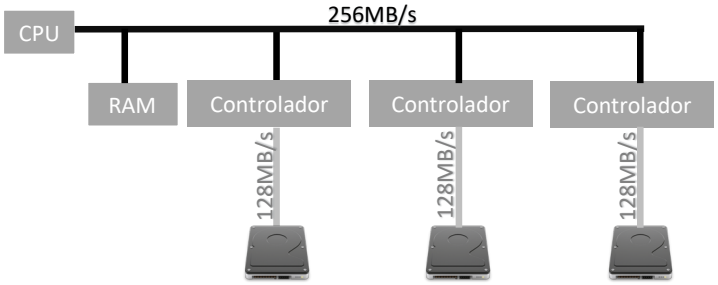
$$\frac{X \times 0{,}5 \times 10^6 \times 2000}{2{,}5 \times 10^9} > 0{,}2$$

$$X > 0{,}5$$

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.

3. 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?

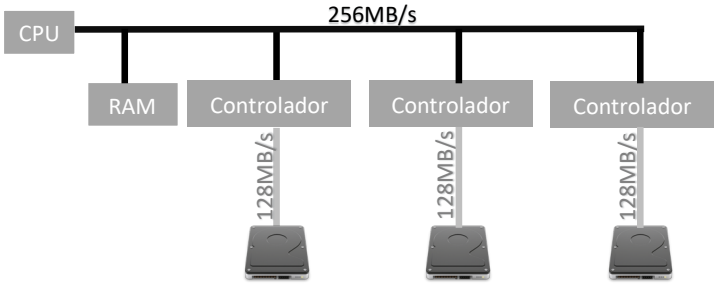
No segundo cenário, o custo (em ciclos de relógio) do *overhead* de cada transferência, incluindo o atendimento da interrupção, é superior ao custo de uma operação de *polling*. A técnica de interrupções é mais vantajosa quando o periférico não está sempre potencialmente ocupado. Para este caso concreto,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

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

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

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

4. 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.

5. 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.

6. 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:

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

7. 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`.

   (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);
```

```
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
```

(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);
```

```
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
```

8. 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.

```
.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
```

9. 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.

```
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
```

[30] 10. O peso ideal de uma pessoa, em função da sua altura e género (feminino ou masculino), pode ser calculado pela seguinte expressão:

$$\text{peso}_{\text{ideal}} = \begin{cases} 72{,}7 \times \text{altura} - 58{,}0 & \text{se género=M} \\ 62{,}1 \times \text{altura} - 44{,}7 & \text{se género=F} \end{cases}$$

Implementar a sub-rotina `pesoideal` que, recebe o peso atual (em kg), a altura (em metros) e o género (`'M'` ou `'F'`) de uma pessoa, calcula a diferença entre o peso atual e o peso ideal. O protótipo da função a invocar em C é:

`float pesoideal(float peso, float altura, char genero).`

Atenção: Deverá implementar a sub-rotina sem recorrer à declaração de constantes.

```
//S0 -> peso
//S1 -> altura
//W0 -> genero
pesoideal:
    MOV W1, 10
    SCVTF S2, W1   //S2 -> 10.0
    CMP W0, 'M'
    BEQ HOMEM
    MOV W1, 621
    MOV W2, 447
    B CPI
HOMEM:
    MOV W1, 727
    MOV W2, 580
CPI:
    SCVTF S3, W1
    FDIV S3, S3, S2 //S3 -> 62.1 ou 72.7
    SCVTF S4, W2
    FDIV S4, S4, S2 //S4 -> 44.7 ou 58.0
    FNMSUB S5, S3, S1, S4
    FSUB S0, S0, S5
  RET
```

11. Pretende-se calcular a distância de um ponto P ao ponto mais distante pertencente a um conjunto de pontos K. Este conjunto, não vazio, é composto por n pontos (x_i, y_i) no plano e é representado por uma sequência de números em memória com a seguinte ordem: $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$.

    A distância de $P = (x, y)$ a um ponto (x_i, y_i) de K é dada por

    $$\sqrt{(x - x_i)^2 + (y - y_i)^2}$$

    Escrever a sub-rotina `Dmax` que calcula a distância máxima de P a um ponto de K utilizando, sempre que possível, instruções SIMD. Considere que `n` é par, `x` e `y` são as coordenadas de P, e K forma um vetor endereçado por `ptK`. O protótipo em C para invocar esta sub-rotina é:

    `float Dmax(float x, float y, float *ptK, int n);`

```
Dmax: FSUB  S2, S2, S2          // Máximo inicial
      LSR   W1, W1, 1           // Número de iterações = n/2
      INS   V0.S[1], V1.S[0]    // V0 = |-|-|y|x|
      INS   V0.D[1], V0.D[0]    // V0 = |y|x|y|x|
L1:   LDR   Q1, [X0], 16        // Lê coordenadas de 2 pontos de K
      FSUB  V1.4S, V0.4S, V1.4S
      FMUL  V1.4S, V1.4S, V1.4S
      FADDP V1.4S, V1.4S, V1.4S
      SMAXV S1, V1.4S
      FCMP  S1, S2
      FCSEL S2, S1, S2, GT      // Atualiza máximo
      SUB   W1, W1, 1
      CBNZ  W1, L1
      FSQRT S0, S2
      RET
```

12. A sub-rotina *assembly* com o protótipo

    `void cypher(unsigned char *clt)`

    cifra um texto apontado por `clt` e terminado pelo código ASCII `0`.

    Esta sub-rotina recorre à sub-rotina de protótipo

    `unsigned char rot13(unsigned char c)`

    que cifra uma letra, trocando-a pela letra que dista dela 13 posições no alfabeto, de forma circular (A → N, B → O, ..., M → Z, N → A, ..., Z → M). Caracteres que não sejam letras maiúsculas no texto original não devem ser cifrados. Por exemplo, a frase `"FEUP 2019 mpcp"` transforma-se em `"SRHC 2019 mpcp"`.

    Nota: As letras maiúsculas estão situadas consecutivamente entre `'A'` (65) e `'Z'` (90).

    (a) Escreva, em *assembly* Aarch64, a sub-rotina `rot13` que recebe uma letra maiúscula e cifra-a pelo processo descrito acima.

    ```
    rot13:
       add  w0,w0,13        // x+13
       cmp  w0,'Z'
       b.le done
       sub  w0,w0,'Z'-'A'+1 // Ajusta no caso de ultrapassar o Z
     done:
       ret
    ```

    (b) Escreva, em *assembly* Aarch64, a sub-rotina `cypher` que altera o texto apontado por `clt` usando a sub-rotina da alínea anterior para cifrar *apenas* as letras maiúsculas, não alterando os restantes caracteres no texto original.

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

    cypher:
      stp   x29,x30,[sp,#-32]!   // Guarda FP e LR. Reserva espaço para mais dois registos

    next:
    ```

```
    ldrb w1,[x0]
  cbz  w1,fim       // Se chega ao fim do texto termina
  cmp  w1,'A'      // Verifica se não
  b.lo fora         // se trata de uma letra
  cmp  w1,'Z'      // maiúscula
  b.hi fora

  str  x0,[sp,16]          // Guarda X0 (apontador para o texto)
  mov  w0,w1                // Argumento de rot13 deve estar em w0
  bl   rot13
  mov  w1,w0                // Resultado (letra cifrada) deve ficar em w1
  ldr  x0,[sp,16]          // Recupera X0
  strb w1,[x0]              // Altera texto

fora:
  add  x0,x0,1             // Próximo caracter
  b    next                 // Repete

fim:
    ldp  x29,x30,[sp],32     // Recupera FP e LR
  ret
```

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

    Ficheiro *main.c*

```c
#include <stdio.h>
extern int sd (unsigned int num);
int main(void)
{
    unsigned int n = 2345, m=567;
    int          r = sd(n) - sd(m);
    printf("r= %d\n",r);
    return 0;
}
```

    Ficheiro *sd.s*

```
1  .text
2  .globl sd
3  .type sd, %function
4
5  sd:   mov   W1, W0
6        mov   W0, 0
7        mov   W2, 10
8  L2:   cbz   W1, L1
9        udiv  W4, W1, W2
10       msub  W5, W4, W2, W1
11       add   W0, W0, W5
12       mov   W1, W4
13       b     L2
14 L1:   ret
```

    Justificar todas as respostas.

(a) Explique como o efeito da instrução `msub` (linha 10) pode ser obtido por uma sequência de duas instruções.

> mul e sub

(b) O bloco de código das linhas 9–10 determina o valor do registo `W5` a partir do valor do registo `W1`. Determinar a relação entre os valores de `W5` e `W1`.

> `W5` contém o resto da divisão de `W1` por 10.

(c) No curso da execução da chamada `sd(n)`, quantas vezes é executada a instrução da linha 13?

> 4 vezes

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

> Logo, a mensagem apresentada é:
>
> `r= -4`

14. 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      beq     L3
9      add     X1, X1, 1
10     ands    X2, X0, 1
11     beq     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.

(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.

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

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

> 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.

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

> 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.

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

> 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
> ```

Fim.