Este exame tem 6 questões, num total de 200 pontos. Fundamente todas as respostas. Fundamentar todas as respostas. Usar uma folha simples apenas para as respostas às perguntas 1, 2 e 3.

Escolha múltipla

1. Indicar na folha de prova a única resposta correta a cada uma das seguintes alíneas.

[8] (a) Considerar o seguinte fragmento de código *assembly*:

```
        .data
v1      BYTE 55H
v2      BYTE F0H
        .code
        xor al, al
        or  al, v1
        xor al, v2
```

O valor final de `AL` é:

A. 55H B. 50H C. 0F5H **D. 0A5H**

[8] (b) Considerar a seguinte sub-rotina.

```
func PROC x:sdword, y:sdword
LOCAL tmp: sdword
...
func ENDP
```

Qual das seguintes instruções é inválida?

A. `mov tmp, y` B. `mov eax, y` C. `mov tmp, ecx` D. `lea eax, tmp`

[8] (c) O disco magnético de um computador é substituído por outro com maior velocidade de rotação, sendo as restantes caraterísticas iguais. Qual dos seguintes fatores é afetado negativamente?

A. débito
B. tempo de resposta
C. consumo de energia
D. capacidade de armazenamento

[8] (d) Qual das seguintes afirmações sobre DMA (acesso direto a memória) é falsa?

A. A utilização de DMA requer um controlador específico para esse fim.
B. Transferências por DMA não requerem o uso de interrupções.
C. A utilização de DMA é mais complicada que a realização de varrimentos *(polling)*.
D. A utilização de DMA é indicada para dispositivos de alta velocidade.

[8] (e) Um sistema RAID-6 usa discos de 0,5 TB. Pretende-se que o sistema tenha uma capacidade útil mínima de 19 TB. Por razões de fiabilidade, cada grupo não deve ter mais de 6 discos no total. Qual é o número mínimo de discos necessário?

A. 65 **B. 60** C. 76 D. 48

Sistemas de entrada/saída

[20] 2. Um disco magnético possui 64 setores de 4 kB por pista e a sua velocidade de rotação é 10000 RPM. Nas condições mais desfavoráveis um ficheiro de 40 kB demora 121 ms a ser lido e 13 ms nas condições mais favoráveis. Calcule o tempo médio de busca e a taxa de transferência desse disco.

Nota: Comece por estabelecer as expressões do tempo de leitura do ficheiro, em função dos parâmetros pedidos, para as duas situações referidas. [Considere kB = 10^3 B, MB = 10^6 B.]

Resposta:

Um ficheiro de 40 kB ocupa 10 setores de 4 kB. As condições mais desfavoráveis correspondem a ter o ficheiro distribuído por 10 setores aleatórios do disco. As condições mais favoráveis correspondem a ter o ficheiro armazenado em 10 setores consecutivos numa única pista.

Assumindo que o *overhead* do controlador é nulo, o tempo de leitura (t_L) do ficheiro completo, nas diferentes condições, será:

$$t_L = \begin{cases} 10 \times \left(t_S + \frac{1}{2} \times \frac{60}{10\times10^3} + \frac{4\times10^3}{T_x}\right) & \text{10 setores aleatórios} \\ t_S + \frac{1}{2} \times \frac{60}{10\times10^3} + 10 \times \frac{4\times10^3}{T_x} & \text{10 setores consecutivos na mesma pista} \end{cases}$$

Resolvendo estas duas equações em simultâneo:

$$\begin{cases} 10t_S + 30\times10^{-3} + \frac{40\times10^3}{T_x} = 121\times10^{-3} \\ t_S + 3\times10^{-3} + \frac{40\times10^3}{T_x} = 13\times10^{-3} \end{cases} \Leftrightarrow \begin{cases} 10t_S + \frac{40\times10^3}{T_x} = 91\times10^{-3} \\ t_S + \frac{40\times10^3}{T_x} = 10\times10^{-3} \end{cases}$$

$$\begin{cases} 9t_S = 81\times10^{-3} \\ - \end{cases} \Leftrightarrow \begin{cases} t_S = 9\times10^{-3} \\ 9\times10^{-3} + \frac{40\times10^3}{T_x} = 10\times10^{-3} \end{cases} \Leftrightarrow \begin{cases} t_S = 9\,\text{ms} \\ T_x = 40\,\text{MB/s} \end{cases}$$

3. Um computador possui um CPU capaz de executar 1000 MIPS. O barramento principal desse computador (FSB) tem uma largura de banda de 800 MB/s. O sistema tem 4 controladores de disco. Cada controlador é capaz de transferir até 200 MB/s e de controlar até 4 discos em simultâneo. O sistema possui um disco magnético de 500 GB instalado em cada controlador, com setores de 4 kB, taxa de transmissão de 80 MB/s, tempo de busca médio de 9 ms e latência de rotação de 0,95 ms. O acesso a um setor gasta 10000 instruções do CPU. [Considere kB = 10^3 B, MB = 10^6 B.]

[10] (a) Indique qual dos componentes fixos (CPU, FSB, controladores de disco) está a limitar o desempenho global do sistema.

Resposta: Comparando os diferentes componentes fixos do sistema em termos de acessos a setores por segundo:

$$N_{CPU} = \frac{1000 \times 10^6}{10 \times 10^3} = 100000; \qquad N_{FSB} = \frac{800 \times 10^6}{4 \times 10^3} = 200000;$$

Débito dos controladores: $4 \times 200\,\text{MB/s} = 800\,\text{MB/s}$ ou seja, semelhante ao FSB.

Conclui-se portanto ser o CPU o elemento limitativo.

[10] (b) Indique se é possível quadruplicar a capacidade de armazenamento do sistema continuando a recorrer a discos de 500 GB e taxa de transmissão 80 MB/s (i) semelhantes aos já instalados, (ii) em tecnologia SSD.

Resposta: Para quadruplicar a capacidade do sistema, recorrendo a discos em tudo semelhantes aos já instalados, será necessário instalar mais 3 discos em cada controlador. O tempo médio de leitura de um setor e o débito médio serão, respetivamente:

$$9 \times 10^{-3} + 0.95 \times 10^{-3} + \frac{4 \times 10^3}{80 \times 10^6} = 10\,\text{ms} \qquad \text{e} \qquad \frac{4\,\text{kB}}{10\,\text{ms}} = 400\,\text{kB/s}$$

logo, o débito médio de 4 discos será 1,6 MB/s, muito inferior ao máximo suportado por cada controlador.

Se forem discos em tecnologia SSD (assumindo que são semelhantes, em todo o resto, aos já instalados) o débito médio de cada disco será 80 MB/s; logo, o débito médio de 4 discos será 320 MB/s, valor superior ao máximo admitido por cada controlador. Concluindo: não é possível quadruplicar a capacidade deste sistema recorrendo a discos SSD (de 500 GB, 80 MB/s) pois haverá perda de informação caso todos os discos de um mesmo controlador funcionem em simultâneo.

Programação

4. Pretende-se imprimir, na forma de gráfico de barras horizontal, o número de ocorrências de cada dígito de uma sequência aleatória de dígitos ASCII de tamanho máximo 50 dígitos e terminada por 0. Exemplo (38 dígitos ASCII):

```
31099581816517957956980691175735181857
==================================================
9: ======
8: =====
7: =====
6: ===
5: =======
4:
3: ==
2:
1: ========
0: ==
==================================================
```

[20] (a) Escrever a rotina `freqdig PROTO Buf:ptr byte, Dig:byte` que conta o número de ocorrências do dígito `Dig` na sequência apontada por `Buf`.

Resposta:

```
freqdig proc uses esi ecx Buf:ptr byte, Dig:byte
      xor ecx,ecx        ;; counter
      mov esi, Buf       ;; start of sequence
next: mov al,[esi]
      and al,al
      jz fim             ;; jump if end of sequence reached
      sub al,'0'         ;; convert from ASCII to binary
      cmp al,Dig
      jnz diff
      inc ecx            ;; found one more!
diff: inc esi
      jmp next
fim:  mov eax,ecx
      ret
freqdig endp
```

[10] (b) Escrever a rotina `pbar PROTO N:byte` que imprime uma barra de sinais '=' de tamanho `N` ($0 \leq$ `N` ≤ 50), recorrendo à variável global pré-inicializada `barra`.

```
barra BYTE  53 dup('='),0
```

Nota: repare que para controlar o tamanho da barra a imprimir basta colocar o terminador de *string* na posição conveniente da variável `barra`.

Resposta:

```
pbar proc uses edi ebx N:byte
      movzx ebx,N
      mov edi,offset barra
      add edi,ebx
      mov byte ptr [edi],0
      invoke printf, offset barra
      mov byte ptr [edi],'='
      ret
pbar endp
```

[10] (c) Completar o programa principal, nos dois locais assinalados com "...", que imprime o gráfico de barras pretendido com o formato indicado no exemplo.

```
include mpcp.inc
;; sub-rotinas definidas noutro ficheiro
freqdig proto Buf:ptr byte, Dig:byte
pbar    proto N:byte
      .data
texto byte  '3109958181651795795698069l175735181857',0
CRLF  byte  10,13,0
barra byte  53 dup('='),0
      ...
```

```
      .code
main: invoke printf, offset texto
      invoke printf, offset CRLF
      invoke printf, offset barra
      invoke printf, offset CRLF
      ...
      invoke printf, offset barra
      invoke _getch
      invoke ExitProcess,0
end main
```

Resposta: A resposta está indicada em fundo cinzento.

```
include mpcp.inc

freqdig proto Buf:ptr byte, Dig:byte
pbar    proto N:byte
;; rotinas definidas num ficheiro separado

      .data
texto byte  '3109958181651795795698069l175735181857',0
CRLF  byte  10,13,0
barra byte  53 dup('='),0

frmt  byte  '%d: ',0

      .code
main: invoke printf, offset texto
      invoke printf, offset CRLF
      invoke printf, offset barra
      invoke printf, offset CRLF

      mov ebx,10
      .repeat
          dec ebx
          invoke printf, offset frmt, ebx
          invoke freqdig, offset texto, bl
          invoke pbar, al
          invoke printf, offset CRLF
      .until(ebx==0)

      invoke printf, offset barra
      invoke _getch
      invoke ExitProcess,0
end main
```

5. O movimento de uma partícula em função do tempo t é definido pela função

$$s(t) = \begin{cases} 2 + t \times \sin(\pi \times t/2) & \text{se} \quad 0 \leq t < 2 \\ 4 \times \dfrac{\cos(t-2)}{t} & \text{se} \quad t \geq 2 \end{cases}$$

[20] (a) Implemente a sub-rotina `dist` que calcula o valor de $s(t)$.

O protótipo da sub-rotina é: `dist proto t:real8`

Resposta:

```
dist proc t:real8
    fld1
    fadd  st(0), st(0)     ; 2.0
    fld   t
    fcomip st(0), st(1)
    jae   L1
    ; Calcula s(t) para t<2
    fldpi
    fmul  t
    fdiv  st(0), st(1)
    fsin
    fmul  t
    fadd
    jmp   L2
L1: ; Calcula s(t) para t>=2
    fld   t                ; st(0)=t e st(1)=2.0
    fsub  st(0), st(1)
    fcos
    fdiv  t
    fmul  st(0), st(1)     ; *2
    fmul                   ; *2 e só st(0) fica não vazio
L2: ret
dist endp
```

[20] (b) A velocidade média de uma partícula no intervalo de tempo $[t_1, t_2]$ define-se como $\overline{v}(t) = \frac{s(t_2)-s(t_1)}{t_2-t_1}$.

Implemente a sub-rotina `velM` que calcula o valor de $\overline{v}(t)$. O protótipo da sub-rotina é: `velM proto t1:real8, t2:real8`

Resposta:

```
velM proc t1:real8, t2:real8
local aux: real8
    invoke dist, t1
    fstp  aux
    invoke dist, t2
    fsub aux
    fld   t2
    fsub  t1
    fdiv
    ret
velM endp
```

Análise de código

6. Considere o seguinte programa constituído por dois ficheiros (`prog.cpp` e `rotinas.asm`).

```
// Ficheiro prog.cpp
extern "C" void newSeq(int *s1, int *s2, int *s, unsigned int nval);
int SEQ1[6]={4, -9, 3, 8, 5, 4},
    SEQ2[6]={2, -1, 3, 0, 1, -3}, S[6];

main ()
{
  newSeq(SEQ1, SEQ2, S, 6);
  for (unsigned int i=0; i<6; i++)
    std::cout << S[i] << '  ';
  return 0;
}
```

```
1  ;; Ficheiro rotinas.asm
2  sumN proc uses ebx esi pt:ptr sdword, N:sdword
3       mov  esi, pt
4       mov  eax, [esi]
5       cmp  N, 0
6       je   L3
7       mov  ecx, N
8       .if  N < 0
9          neg  ecx
10 L1:     sub  esi, 4
11         add  eax, [esi]
12         loop L1
13      .else
14 L2:     add  esi, 4
15         add  eax, [esi]
16         loop L2
17      .endif
18 L3:  ret
19 sumN endp
20
21 newSeq proc uses esi edi ebx pt1:ptr sdword, pt2:ptr sdword,
22                                                pt:ptr sdword, N:dword
23      mov  ecx, N
24      mov  esi, pt1
25      mov  ebx, pt2
26      mov  edi, pt
27      .while ecx > 0
28         push ecx
29         invoke sumN, esi, sdword ptr [ebx]
30         mov  [edi], eax
31         add  esi, 4
32         add  ebx, 4
33         add  edi, 4
34         pop  ecx
35         dec  ecx
36      .endw
37      ret
38 newSeq endp
```

[10] (a) Descreva a função realizada pela sub-rotina `sumN`.

Resposta:

A sub-rotina `sumN` copia para `EAX` o elemento de uma sequência apontado por `pt`. Se `N`≠0, os |`N`| elementos seguintes (se `N`>0) ou anteriores (se `N`<0) são somados ao valor de `EAX`.

[10] (b) Indique e explique o que é apresentado no monitor quando o programa é executado.

Resposta:

O programa imprime a sequência de números inteiros `S` obtida através da sub-rotina `newSeq`. Cada elemento $e_i^{\langle S \rangle}$ desta sequência é calculado pela sub-rotina `sumN` a partir dos elementos de `SEQ1` e `SEQ2` na mesma posição correspondente.

$$e_i^{\langle S \rangle} = \begin{cases} e_i^{\langle S1 \rangle} & \text{se} \quad e_i^{\langle S2 \rangle} = 0 \\ \displaystyle\sum_{k=i}^{i+e_i^{\langle S2 \rangle}} e_k^{\langle S1 \rangle} & \text{se} \quad e_i^{\langle S2 \rangle} \neq 0 \end{cases}$$

($e_i^{\langle X \rangle}$ denota um elemento da sequência `X`)

No monitor aparece: -2 -5 20 8 9 20

[10] (c) Admita que `SEQ1` passa a ter um sétimo elemento igual a -1. Nestas condições, sabendo que após a sétima execução de `sumN` resulta `EAX`=8, determine qual deverá ser o sétimo elemento de `SEQ2`.

Resposta:

O valor de `EAX` resulta da soma de -1 com elementos anteriores. Para ser `EAX`=8 os elementos anteriores a somar são o 4 e o 5. Logo, o sétimo elemento de `SEQ2`, que indica o número de elementos anteriores a somar a -1, deverá ser -2.

[10] (d) Da primeira vez que o prólogo de `sumN` é executado resulta o conteúdo da pilha apresentado na tabela. Determine os endereços, inicial e final, das posições de memória ocupadas pela sequência `SEQ1` declarada em `main()`.

Endereço (hex.)	Conteúdo (hex.)
004CFA0C	004CFA2C
004CFA08	70881022
004CFA04	7088100A

Resposta: O prólogo de `sumN` é:

```
push ebp
mov  ebp, esp
push ebx
push esi
```

Assim, no topo da pilha (endereço `004CFA04H`) está o conteúdo de `ESI` e no endereço seguinte está o conteúdo de `EBX`. O conteúdo de `ESI` e `EBX` é o endereço inicial de, respetivamente, `SEQ1` e `SEQ2` (após invocação de `newSeq` ver as linhas de código 24, 25, 29 e 2).

Sendo o conteúdo da pilha apresentado referente à primeira execução do prólogo, então `7088100AH` é o endereço inicial de `SEQ1` e `70881022H` é o endereço inicial de `SEQ2`. O último elemento de `SEQ1` ocupa os 4 bytes anteriores a `70881022H`.

Portanto, `SEQ1` ocupa os endereços `7088100AH` a `70881021H`.

Alternativamente, conhecido o endereço inicial de `SEQ1` e tendo em consideração que esta possui 6 elementos, o endereço do primeiro byte do último elemento é `7088100AH`$+(6-1)\times 4 =$`7088101EH`. Cada elemento ocupa 4 bytes, pelo que o endereço final de `SEQ1` é `7088101EH`$+3 =$`70881021H`.

Fim do enunciado.