Este exame tem 6 questões, num total de 200 pontos. Fundamentar todas as respostas.
Responder às questões 1, 2 e 3 na mesma folha simples; responder a cada uma das restantes numa folha simples.

Escolha múltipla

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

[8] (a) Observou-se que um ficheiro que ocupa vários setores de um disco demora sempre o mesmo tempo a ser lido, independentemente de residir em setores aleatórios ou consecutivos. À luz desta informação, qual das seguintes afirmações é verdadeira?

A. Trata-se de um disco magnético.

B. O ficheiro ocupa mais do que uma pista.

C. Trata-se de um disco *Solid State* (SSD).

D. O ficheiro ocupa menos do que uma pista.

[8] (b) Um disco possui as seguintes características: setores de 4 kB, taxa de transmissão de 4000 kB/s, tempo médio de busca mais latência rotacional de 9 ms. Em média, quantos bytes de informação consegue debitar por segundo?

A. 400k B. 1000k C. 100k D. 4000k

[8] (c) Numa sub-rotina com argumentos de entrada, registos a preservar e variáveis locais, em que momento é reservado o espaço para as variáveis locais?

A. Imediatamente depois de serem guardados os argumentos de entrada.

B. Imediatamente antes de serem preservados os registos.

C. Imediatamente depois de serem preservados os registos.

D. Imediatamente antes de ser guardado o endereço de retorno.

[8] (d) Assumindo que AL=58h, qual das seguintes instruções tem como resultado AL=85h?

A. `ror AL, 4`

B. `xor AL, 85h`

C. `rcr AL, 4`

D. `xor AL, 58h`

[8] (e) A sequência de instruções `PUSH EBX / CALL 1C36AB65h` corresponde à invocação completa de uma determinada sub-rotina. Qual das seguintes afirmações pode ser justificada pela informação fornecida?

A. A sub-rotina guarda o registo EBX no prólogo (`uses EBX...`).

B. A sub-rotina utiliza uma variável local de 32 bits.

C. A sub-rotina utiliza um total de 32 bits de variáveis locais.

D. A sub-rotina tem um parâmetro de entrada.

Sistemas de entrada/saída

2. Um periférico de computador transfere dados à taxa média de 2 kB/s e máxima de 6 kB/s. Cada acesso ao periférico transfere 8 bytes. Esse periférico é gerido por recurso a interrupções e cada execução da sub-rotina de atendimento necessita de 10^5 ciclos. O CPU funciona a 1 GHz. [Assumir kB=10^3 B, MB=10^6 B, GB=10^9 B.]

[10] (a) Determinar a percentagem de tempo que, em média, o CPU está ocupado a atender o periférico.

Resposta: Número de operações por segundo:

$$\frac{2\,\text{kB/s}}{8\,\text{B/IOP}} = 250\,\text{IOP/s}$$

Número de ciclos despendidos: $250\,\text{IOP/s} \times 10^5\,\text{ciclos/IOP} = 2{,}5 \times 10^7\,\text{ciclos/s}$

Percentagem de tempo:

$$\frac{2{,}5 \times 10^7}{10^9} = 0{,}025 = 2{,}5\,\%$$

[10] (b) Uma versão anterior do sistema usa *varrimento* para gerir o mesmo periférico. A execução da sub-rotina de varrimento toma 5×10^4 ciclos. Determinar a taxa de ocupação do CPU nessa situação.

Resposta: É necessário usar a taxa máxima para calcular o número de operações:

$$\frac{6\,\text{kB/s}}{8\,\text{B/IOP}} = 750\,\text{IOP/s}$$

Portanto, o número de ciclos despendidos é:

$$750\,\text{IOP/s} \times 5 \times 10^4\,\text{ciclos/IOP} = 375 \times 10^5\,\text{ciclos/s} = 3{,}75 \times 10^7\,\text{ciclos/s}$$

Percentagem de tempo: $\frac{3{,}75 \times 10^7}{1 \times 10^9} = 0{,}0375 = 3{,}75\,\%$.

3. Uma placa de rede para comunicação sem fios usada num portátil pode transmitir à taxa máxima de 160 kB/s. O consumo de energia da placa de rede é de 20 J/s.

   A bateria de um portátil tem uma capacidade máxima de $3{,}6 \times 10^5$ J. O conjunto de todos os outros elementos do portátil (exceto a placa de rede) consome 60 J/s.

   O disco magnético do portátil funciona a 2000 RPM, o tempo de busca médio é 9 ms e a taxa de transferência é de 4 MB/s. Ignorar *overhead* do controlador.

   Para executar uma dada tarefa, o portátil precisa de receber N blocos de dados, que armazena no disco magnético. Cada bloco tem 16 kB e é guardado em setores consecutivos.

Blocos diferentes estão dispostos aleatoriamente no disco.

[5] (a) Assumindo que a bateria está totalmente carregada, determinar o tempo de funcionamento do portátil, se não for usada a placa de rede.

Resposta:

$$\frac{3{,}6 \times 10^5\,\mathrm{J}}{60\,\mathrm{J/s}} = \frac{36 \times 10^3}{6}\,\mathrm{s} = 6 \times 10^3\,\mathrm{s} \qquad (100 \text{ minutos})$$

[8] (b) Determinar o tempo que demora a armazenar um bloco de dados.

Resposta: O tempo de busca médio é conhecido. Falta determinar a latência de rotação e o tempo de transferência.

$$t_{\mathrm{rot}} = 0{,}5 \times \frac{60}{2000} = 15 \times 10^{-3}\,\mathrm{s} = 15\,\mathrm{ms}$$

$$t_{\mathrm{transf}} = \frac{16\,\mathrm{kB}}{4\,\mathrm{MB/s}} = 4\,\mathrm{ms}$$

O tempo para gravar um bloco de dados é $9\,\mathrm{ms} + 4\,\mathrm{ms} + 15\,\mathrm{ms} = 28\,\mathrm{ms}$.

[7] (c) Determinar o valor máximo de N, assumindo que receção via rede e armazenamento em disco são feitos consecutivamente (por blocos) e sem sobreposição. A energia disponível na bateria é $2{,}5 \times 10^5$ J.

Resposta: Tempo para receber 1 bloco (rede): $\frac{16\,\mathrm{kB}}{160\,\mathrm{kB/s}} = 100\,\mathrm{ms}$.
Energia necessária para receber um bloco:

$$E_{\text{receção}} = (20 + 60)\mathrm{J/s} \times 0{,}1\,\mathrm{s} = 8\,\mathrm{J}$$

Energia necessária para armazenar o bloco (placa de rede está desligada):

$$E_{\text{armazenamento}} = 60\,\mathrm{J/s} \times 0{,}028\,\mathrm{s} = 1{,}68\,\mathrm{J}$$

Assim, o processamento total de um bloco consome $E_{\text{bloco}} = 8\,\mathrm{J} + 1{,}68\,\mathrm{J} = 9{,}68\,\mathrm{J}$.

$$N_{\max} = \left\lfloor \frac{2{,}5 \times 10^5\,\mathrm{J}}{9{,}68\,\mathrm{J}} \right\rfloor$$

Programação

4. Podem implementar-se operações de deslocamento e rotação de sequências de `nelem` elementos semelhantes às existentes para registos.

[20] (a) Pretende-se deslocar os elementos de uma sequência `N` posições para a direita (com 0<`N`<`nelem`), mantendo cópia do primeiro elemento nas `N` posições iniciais.

Exemplo: para a sequência {5, 0, -3, 8, -2, -1, 3}, o resultado do deslocamento do conteúdo três posições para a direita é {5, 5, 5, 5, 0, -3, 8}.

Implemente a sub-rotina `shfRight` que realiza a operação de deslocamento sobre a sequência endereçada por `pt`. O protótipo da sub-rotina é:

```
shfRight proto pt:ptr sdword, nelem:dword, N:dword
```

Resposta:

```
shfRight proc uses esi edi pt:ptr sdword, nelem:dword, N:dword
    ; Copia os primeiros nelem-N elementos para a direita,
    ; começando pelo último.
    std
    mov   esi, nelem
    sub   esi, N
    dec   esi
    shl   esi, 2     ; Multiplica por 4
    add   esi, pt    ; Origem
    mov   edi, N
    shl   edi, 2
    add   edi, esi   ; Destino
    mov   ecx, nelem
    sub   ecx, N
    rep movsd
    ; Replica o primeiro elemento nas N posições da esquerda.
    mov   esi, pt
    mov   eax, [esi]
    mov   ecx, N
    dec   ecx
    rep stosd        ; EDI aponta posição que recebe réplica
    cld
    ret
shfRight endp
```

[20] (b) Pretende-se rodar os elementos de uma sequência `N` posições para a esquerda (com 0<`N`<`nelem`).

Exemplo: para a sequência {5, 0, -3, 8, -2, -1, 3}, o resultado da rotação do conteúdo três posições para a esquerda é {8, -2, -1, 3, 5, 0, -3}.

Implemente a sub-rotina `rotLeft` que preenche uma nova sequência, endereçada por `newpt`, com os elementos resultantes da rotação da sequência endereçada por `pt`. Esta sub-rotina não deve ter ciclos explícitos, mas pode conter instruções dedicadas para tratamento de sequências. O protótipo da sub-rotina é:

```
rotLeft proto pt:ptr sdword, newpt:ptr sdword, nelem:dword, N:dword
```

Resposta:

```
rotLeft proc uses esi edi pt:ptr sdword, newpt:ptr sdword,
                                         nelem:dword, N:dword
    ; Copia os últimos nelem-N elementos
    mov   esi, pt
    mov   edi, N
    shl   edi, 2   ; Multiplica por 4
    add   esi, edi ; End. do elemento que ocupará a 1ª posição.
    mov   edi, newpt
    mov   ecx, nelem
    sub   ecx, N
    rep movsd
    ; Copia os primeiros N elementos
    mov   esi, pt
    mov   ecx, N
    rep movsd
    ret
rotLeft endp
```

5. Um *drone* é usado para transportar cargas de uma margem de um rio para a outra. A carga máxima em cada viagem é de 15 kg.

   A sequência P (de números reais) contém o peso (em kg) de N itens a transportar ($N > 0$). Os itens são colocados no *drone* pela ordem em que estão na sequência.

[20] (a) Desenvolver uma sub-rotina para determinar o número de viagens necessárias para transportar todos os itens. (Não incluir as viagens de retorno.) O resultado da sub-rotina é do tipo `DWORD`.

Exemplo: Para $P = \{11.5, 3.1, 2.5, 14.1, 4.5, 5.6\}$, os pesos transportados em cada viagem são 14.6, 2.5, 14.1 e 10.1, para um total de 4 viagens.

A sub-rotina deve ter o protótipo: `nviagens PROTO P: ptr real8, N: dword`.

Resposta:

```
      .data
QUINZE REAL8 15.0
      .code
nviagens PROC  P:ptr real8, N:dword
         mov ecx, N
         mov edx, P
         xor eax, eax
         fldz
ciclo:   fadd REAL8 PTR [edx]
         fld  QUINZE
         fcomip st(0),st(1)
         jae  L1
         inc  eax
         fsub st(0), st(0) ; recolocar a zero
         fadd REAL8 PTR [edx]  ; novo peso inicial
L1:      add  edx, 8
```

```
        loop ciclo
        fstp st(0)      ; limpar pilha UVF
        inc eax
        ret
nviagens ENDP
```

[20] (b) Desenvolver uma sub-rotina para determinar quanto da capacidade de transporte é desperdiçada. Esta sub-rotina deve invocar `nviagens`. Assumir que está disponível a sub-rotina `soma PROTO P:ptr real8, N:dword` que calcula a soma dos `N` elementos de uma sequência `P` de números reais.

Exemplo: No caso do exemplo, 4 viagens permitiriam transportar 60 kg, mas apenas 41,3 kg são de facto transportados. Logo, desperdiçaram-se 18,7 kg.

A sub-rotina deve ter o protótipo: `desperdicio PROTO P:ptr real8, N:dword`.

Resposta:

```
desperdicio PROC P:ptr real8, N:dword
        LOCAL aux:DWORD     ; guardar número de viagens
        LOCAL total: REAL8
        invoke soma, P, N
        fstp total
        invoke nviagens, P, N
        mov aux,eax
        fild aux
        fmul QUINZE
        fsub total              ; resultado em st(0)
        ret
desperdicio ENDP
```

Análise de código

6. Considere o seguinte programa composto por dois ficheiros.

```
1  // Ficheiro main.cpp
2  #include <iostream>
3  extern "C" void calc(unsigned int *seq, unsigned int N,
4                       unsigned int *res, unsigned int v);
5  unsigned int valores [7] = {23, 71, 24, 20, 3, 5, 8};
6  unsigned int resultado[5];
7
8  int main()
9  {
10         calc(valores, 7, resultado, 5);
11         for (int i = 0; i < 5 ; i++)
12                 std::cout << resultado[i] << ' ';
13         return 0;
14 }
```

```
1  ;; Ficheiro calc.asm
2  include mpcp.inc
```

```
3  .code
4  aux PROC C val:dword, d: dword
5      mov eax, val
6      mov ecx, d
7      .WHILE eax >= d
8          sub eax, d
9      .ENDW
10     ret
11 aux ENDP
12
13 calc PROC C USES esi edi seq: ptr dword, N:dword,
14                          res:ptr dword, v: dword
15     xor eax, eax
16     mov edi, res
17     mov ecx, v
18     rep stosd
19
20     mov edi, seq
21     mov esi, res
22     mov ecx, N
23 L1: push ecx
24     invoke aux, [edi], v
25     inc dword ptr [esi+4*eax]
26     add edi, 4
27     pop ecx
28     loop L1
29     ret
30 calc ENDP
31 end
```

[10] (a) Considerar a primeira vez que a sub-rotina `aux` é executada. Determinar o número de vezes que é executada a instrução `sub` e qual é o valor final do registo `eax`.

> **Resposta:** Para a primeira invocação, `val` é 23 e `d` é 5 (o mesmo valor para todas as invocações). O registo `eax` tem sucessivamente os valores 23, 18, 13, 8 e 3. A instrução `sub` é executada 4 vezes e o valor final de `eax` é 3 (i.e., o resto da divisão de 23 por 5).

[10] (b) Explicar para que efeito é utilizada a sub-rotina `aux` e determinar quantas vezes é invocada durante a execução do programa.

> **Resposta:**
> A sub-rotina `aux` calcula o resto da divisão de `val` por `d` por subtração sucessiva de `d` até que o valor resultante fique abaixo de `d`. A sub-rotina é invocada a partir de `calc` uma vez para cada elemento de `seq`. Portanto é executada 7 vezes.

[10] (c) Indique o que é apresentado no monitor quando o programa é executado e descreva a função realizada pela sub-rotina `calc`.

Resposta:

A sub-rotina `calc` determina, para cada elemento de `seq`, o resto da sua divisão por `v` e contabiliza quantas vezes ocorre cada valor do resto. Como `v` é 5, os valores possíveis para o resto são 0, 1, 2, 3 e 4. O número de ocorrências correspondente é guardado em posições sucessivas da sequência `res`. Os valores iniciais desta `seq` são colocados a zero pela instrução `rep stosd`.

No monitor aparece o conteúdo da sequência com o número de ocorrências de cada resto: `2 1 0 3 1`.

[10] (d) Assumir a seguinte informação:

endereço inicial de `valores`	`0081F000H`
endereço inicial de `resultado`	`0081F248H`
endereço da instrução na linha 25 de `calc.asm`	`00811DABH`
o código da instrução `add esp, 8` ocupa 3 bytes	

Quando é executada a instrução da linha 6 do ficheiro `calc.asm` pela primeira vez, o valor do registo `ESP` é `0019F700H`. Indicar o endereço e conteúdo dos 5 itens colocados no topo da pilha.

Resposta: A tabela seguinte apresenta o conteúdo e endereço de cada elemento pedido. A última parte da tabela é necessária para determinar o conteúdo do topo da pilha (anterior valor de `EBP` = posição na pilha do valor de `EBP` anterior a esse).

Para determinar o endereço de retorno (segunda linha da tabela) é preciso ter em conta que se está a usar a convenção `C`. Neste caso, a invocação da sub-rotina `aux` na linha 24 termina com:

```
call aux
add  esp, 8
```

O endereço de retorno é o endereço da instrução `add esp, 8`. Como esta instrução ocupa 3 bytes, o seu endereço é `00811DABH-03H = 00811DA8H`.

Endereço	Conteúdo	Significado
`0019F700H`	`0019F71CH`	Valor anterior de `EBP` (ver fim da tabela)
`0019F704H`	`00811DA8H`	Endereço de retorno a `calc`
`0019F708H`	`23`	primeiro valor de `seq`
`0019F70CH`	`5`	Valor de `v`
`0019F710H`	`7`	valor de `ecx` (= `N`)
`0019F714H`	??	valor anterior de `EDI`
`0019F718H`	??	valor anterior de `ESI`
`0019F71CH`	??	valor de `EBP` antes de entrar em `calc`

Fim do enunciado.