Este exame tem 7 questões, num total de 200 pontos. Responda em folhas separadas a cada um dos seguintes conjuntos de problemas: (1 e 2), (3 e 4), (5 e 6). O problema 7 deve ser respondido na folha de enunciado.

1. Considerar o seguinte programa, composto por dois ficheiros.

   Ficheiro `main.cpp`:

```
#include <iostream>

extern "C" int func(unsigned int A[], unsigned int nA,
                    int B[], unsigned int nB);

unsigned  int ind[4]  = {2, 11, 0, 3 };
int           vals[9] = {2, 3, -5, -6, 2, -2, 5, 1, 5};

int main()
{
   std::cout << func(ind, 4, vals, 9) << std::endl;
   return 0;
}
```

   Ficheiro `func.asm`:

```
1  include mpcp2.inc
2
3         .code
4
5  func PROC C USES esi edi ebx A:ptr dword,
6        nA:dword , B:ptr sdword, nB:dword
7
8     LOCAL contador: dword
9
10    mov   esi, A
11    mov   ecx, nA
12    mov   edi, B
13    mov   edx, nB
14    xor   eax, eax
15    mov   contador, eax
16
17    jecxz fim
18
19 @@: mov ebx, [esi]
20    .IF ebx >= 0 && ebx < edx
21       push edi
22       push ebx
23       shl ebx, 2
24       add edi, ebx
25       add eax, [edi]
26       pop ebx
27       pop edi
28       inc contador
29    .ENDIF
30    add esi, 4
31    loop @B
32    cdq
33    idiv contador
34 fim: ret
35 func ENDP
36 END
```

[10] (a) Considere o corpo do ciclo composto pelas linhas 19 a 31.

Explicar *fundamentadamente* a finalidade de uma única iteração do ciclo.

Resposta: O ciclo começa por colocar em `ebx` um valor da sequência `A` (o endereço desse valor está em `edi`). Se o valor de `ebx` estivar na gama [0; comprimento da sequência `B`[(excluindo este extremo), é usado como índice da sequência `B` para calcular o endereço do (`ebx`-1)-ésimo elemento dessa sequência: é multiplicado por

4 e somado ao valor de `edi` (endereço-base da sequência `B`). O elemento da sequência `B` guardado nesse endereço é acumulado em `eax`.

Estas operações estão rodeadas por instruções `push` e `pop` para garantir que o valor de `edi` não é alterado.

A variável local `contador` é incrementada: serve para contabilizar o número de valores acumulados. O valor de `esi` é incrementado de 4 unidades para apontar para o próximo elemento da sequência `A`.

Resumindo: são acumulados em `eax` os valores da sequência `B` que estão nas posições definidas pelo elementos da sequência `A` (desde que estes valores sejam legais).

[10] (b) Para os dados de entrada indicados no ficheiro `main.cpp`, quantas vezes é executada a instrução da linha 25? Justifique.

Resposta: A instrução da linha 25 é executada para todos os valores de `A` que constituam índices da sequência `B`, que neste caso são os valores [0;8] (já que a sequência `B` corresponde ao vetor `vals` do código C++). A sequência `A` corresponde ao vetor `ind` do código C++. Três elementos deste vetor são índices legais (o único valor não-legal é o valor 11). Portanto, a instrução é executada 3 vezes.

[10] (c) Indicar, justificando, o que é apresentado no monitor quando este programa é executado.

Resposta: O programa apresenta o valor retornado por `func` com `ind` como sequência `A` e `vals` como sequência `B`. Conforme indicado na alínea (a), a repetição do ciclo existente em `func` acumula os valores de `vals[2]`, `vals[0]` e `vals[3]` (deixando `eax` com o valor $-5 + 2 - 6 = -9$).

A seguir ao ciclo, o valor de `eax` é dividido pelo número de valores acumulados (neste caso, 3), ficando `eax` com o valor -3 (valor retornado por `func`). Logo, o resultado da função é a parte inteira da média dos valores de `B` indexados (legalmente) por elementos de `A`. Neste caso, o que é apresentado no monitor é o valor **`-3`**.

[10] (d) A sequência de instruções composta pelas linhas 21 a 27 pode ser substituída por uma única instrução. Indique qual e justifique.

Resposta: A instrução equivalente é `add eax [edi+4*ebx]`. As instruções 23 e 24 calculam o valor `edi + 4 * edx`, que é depois usada como endereço pela instrução 25. A utilização das instruções `push` e `pop` garante que os valores finais de `edi` e `ebx` são os mesmos que os iniciais (i.e., não são alterados). A instrução equivalente indicada também não altera os valores de `edi` e `ebx`. Em ambos os casos, o valor do indicadores *(flags)* é determinado pelo valor acumulado em `eax`.

[25] 2. Considere o seguinte fragmento de código de um programa:

```
  .data
seq dword 12, 8, 1, 10, 11
  .code
rot proc uses ebx val1:dword, val2:dword
  ...
rot endp
```

Assuma que no mesmo programa é executada a seguinte sequência de instruções e que, imediatamente antes, `ESP=003F1198h`.

```
mov  ebp, esp
mov  esi, offset seq
mov  ebx, 16
invoke rot, dword ptr [esi], ebx
xor  ebx, ebx
```

O código da instrução `xor ebx,ebx` ocupa o endereço `09AA0015h`.

Mostre o estado da pilha após a execução do prólogo de `rot` indicando o endereço e o conteúdo de cada posição em hexadecimal.

Resposta: A sub-rotina `rot` possui dois parâmetros, os quais são escritos na pilha quando `rot` é invocada. O valor de EBX (`00000010h`) ocupa a posição `003F1194h`.

A posição seguinte (`003F1190h`) é ocupada pelo primeiro elemento de `seq` (`0000000Ch`). A seguir é guardado na pilha o endereço de retorno, ou seja, o endereço da instrução `xor`.
No prólogo é escrito na pilha o valor de `EBP`, igual ao `ESP` inicial (003F1198h) em consequência do `mov`. Por fim, na posição `003F1184h` é guardado o valor `00000010h` dado que `rot` preserva `EBX`.

Endereço	Conteúdo
003F1194	00000010
003F1190	0000000C
003F118C	09AA0015
003F1188	003F1198
003F1184	00000010

[20] 3. Um disco magnético que roda a 10000 RPM tem 516 setores (de 0,5 kB) por cilindro. O seu tempo de busca mínimo é 4 ms e o tempo de busca médio é 10 ms. Sabe-se ainda que a taxa de transferência é de 100 MB/s e a latência do controlador é de 5 ms. Calcule o tempo necessário para transferir um ficheiro de 600 kB, assumindo que os respetivos setores estão dispostos em disco da maneira mais favorável possível (inicialmente, a cabeça de leitura está numa posição aleatória).

Resposta:

O ficheiro ocupa $\frac{600\text{kB}}{0,5\text{kB}} = 1200$ setores. Uma vez que cada cilindro tem uma capacidade de 516 setores, o ficheiro está distribuído por 3 cilindros (516 + 516 + 168) o que implica duas mudanças de cilindro durante a transferência.

T = tempo de busca médio + tempo de rotação + tempo de transferência + tempo de mudança de cilindro + latência do controlador

$$T = 10\text{ms} + 0{,}5 \times \frac{60}{10000} + \frac{600\text{kB}}{100\text{MB/s}} + 2 \times 4\text{ms} + 5\text{ms}$$

$$T = 10\text{ms} + 3\text{ms} + 6\text{ms} + 8\text{ms} + 5\text{ms}$$

$$T = 32\text{ms}$$

[20] 4. Utilizando um computador portátil, pretende-se publicar no "youvideo" 50 vídeos de cerca de 200 MB através de uma ligação sem fios (usando uma *pen* de banda larga) com uma capacidade de *upload* de 4 Mbit/s. Inicialmente, a energia armazenada na bateria do portátil totaliza 1000 kJ. Sabe-se ainda que a *pen* de banda larga consome 11 W quando está a realizar o *upload* de ficheiros e 6 W no restante tempo, enquanto os restantes recursos do portátil consomem 50 W durante todo o tempo que o computador está ligado. Tendo em conta que o processo de publicação de cada vídeo requer 90 s para preenchimento do formulário (título, descrição, etc) e 10 s para a seleção do vídeo a enviar, calcule quantos vídeos se consegue publicar no "youvideo" antes do computador ficar sem energia. Nota: 1 W = 1 J/s.

Resposta:

O tempo de upload de um vídeo é:

$$T_{upload} = \frac{200 \times 8 \times 10^6}{4 \times 10^6} = 400\,\text{s}$$

O tempo de preparação (preechimento do formulário mais seleção) de um vídeo é:

$$T_{prep} = 90\,\text{s} + 10\,\text{s} = 100\,\text{s}$$

O tempo total para enviar um vídeo será então:

$$T_{total} = T_{prep} + T_{upload} = 100\,\text{s} + 400\,\text{s} \ = 500\,\text{s}$$

Este tempo leva a um gasto de energia por ficheiro de:

$$E_{prep} = 100\,\text{s} \times (50\,\text{W} + 6\,\text{W}) = 100\,\text{s} \times 56\,\text{W} = 5600\,\text{J}$$

$$E_{upload} = 400\,\text{s} \times (50\,\text{W} + 11\,\text{W}) = 400\,\text{s} \times 61\,\text{W} = 24400\,\text{J}$$

$$E_{total} = E_{prep} + E_{upload} = 5600\,\text{J} + 24400\,\text{J} = 30000\,\text{J}$$

Como inicialmente a bateria do computador tinha 1000 kJ,

$$\frac{1000000\,\mathrm{J}}{30000\,\mathrm{J}} = 33{,}33$$

o que significa que se consegue publicar 33 vídeos no "youvideo" antes de o computador ficar sem energia.

[25] 5. Considere uma sequência de valores inteiros referentes à temperatura expressa em graus Fahrenheit. Pretende-se preencher uma nova sequência com os valores correspondentes em graus Celsius e calcular o respetivo valor médio. A relação entre os valores da temperatura T_F e T_C nas escalas Fahrenheit e Celsius, respetivamente, é $T_C = \frac{T_F - 32}{1.8}$.

Implemente a sub-rotina `F2Cmed` que a partir do endereço base `ptF` da sequência de `N` inteiros (`N` > 0), preenche a sequência de valores em graus Celsius com endereço base em `ptC` e calcula o valor médio.

O protótipo da sub-rotina é: `F2Cmed proto ptF:ptr sword, ptC:ptr real8, N:dword`

Resposta:

```
    .data
c32  sword 32
c1v8 real8 1.8

    .code
F2Cmed PROC uses esi edi ptF:ptr sword, ptC:ptr real8, N:dword
    mov   esi, ptF
    mov   edi, ptC
    mov   ecx, N
    fldz           ; Inicializa soma de valores
@@: fild  sword ptr [esi]  ; Valor em graus F
    fisub c32
    fdiv  c1v8
    fst   real8 ptr [edi]  ; Valor em graus C
    fadd                   ; ST(0) = soma
    add   esi, 2
    add   edi, 8
    loop  @B
    fidiv N
    ret
F2Cmed endp
```

6. Cada uma das seguintes questões tem apenas uma resposta certa. Indique as respostas corretas **na folha de resposta** (e não na folha do enunciado).

[5] (a) Considere um sistema RAID-5. Um bloco de dados tem o valor `2A7Fh` e o correspondente bloco de paridade tem o valor `0F45h`. O conteúdo do bloco de dados é alterado para `05AFh`. Qual é o novo valor do bloco de paridade?

A. `2FD0h` **B.** `2095h` C. `205Fh` D. `25F5h`

[5] (b) Qual das seguintes instruções é ilegal?

A. `lea [edi+edx], eax` B. `sub [edi+ebx], ebx`
C. `cwde` D. `inc byte ptr [eax+8*ecx]`

[5] (c) Considere que `ST(0)=2.5` e `ST(1)=1.0`. Os restantes registos da UVF estão vazios. Após execução da instrução `FSUBR` resulta:

A. `ST(0)=2.5` e `ST(1)=-1.5` B. `ST(0)=-1.5` e `ST(1) vazio`
C. `ST(0)=1.5` **e** `ST(1) vazio` D. `ST(0)=vazio` e `ST(1)=-1.5`

[5] (d) Indique qual é a definição válida para uma sequência de dados chamada `myArray` e contendo os valores decimais 10, 20 e 30.

A. `BYTE myArray 10, 20, 30` **B.** `myArray BYTE 10, 20, 30`
C. `BYTE myArray[3]: 10, 20,30` D. `myArray BYTE DUP (3) 10,20,30`

[5] (e) Qual é o valor final do registo `AL` após a execução do seguinte código:

```
mov al,3Ch
or  al,82h
```

A. `3Eh` B. `BCh` **C.** `BEh` D. `3Ch`

[5] (f) Assumir que o registo `AL` contém o valor 32. Qual deve ser o conteúdo do registo `BL`, para que a excução da instrução `imul BL` coloque o indicador *(flag)* `OF=1`?

A. 2 **B. 4** C. -4 D. -3

7. Pretende-se determinar e imprimir o número de múltiplos de n, para todos os valores de n entre 2 e 9 (inclusive), existentes numa sequência de inteiros sem sinal terminada pelo valor zero. Se não existirem múltiplos para um dado valor de n, nada é impresso relativamente a esse n.

   Exemplo: para a sequência {21,33,12,17,0} o resultado a imprimir será:

```
21 33 12 17
M2:1 M3:3 M4:1 M6:1 M7:1
```

[18] (a) Completar o programa principal. O programa imprime a sequência (sem o terminador) e na linha seguinte o resultado da análise, no formato indicado acima. A rotina `multn` retorna o número de múltiplos de `n` na sequência apontada por `ptseq`.

(continua na página seguinte)

```
include mpcp.inc
;; sub-rotina definida noutro ficheiro
multn proto n:dword, ptseq:ptr dword

      .data
lista dword 21,33,12,17,0
fmt1  byte  '%d ',0
fmt2  byte  'M%d:%d ',0
crlf  byte  10,13,0

      .code
main: mov esi, offset lista
nxt:  lodsd
      and eax,eax
      jz  fseq
      invoke printf,offset fmt1,eax
      jmp nxt
fseq: invoke printf,offset crlf
      mov ebx,2
      .repeat
        invoke multn, ebx, offset lista
        .if (eax != 0)
          invoke printf,offset fmt2,ebx,eax
        .endif
        inc ebx
      .until(ebx==10)
      invoke _getch
      invoke ExitProcess,0
end main
```

[22] (b) Escrever a rotina `multn PROTO n:dword, ptseq:ptr dword` que retorna o número de múltiplos de `n` existentes na sequência apontada por `ptseq`.

Resposta:

```
multn proc uses esi     n:dword, ptseq:ptr dword
      xor ecx,ecx       ;; reset counter
      mov esi, ptseq    ;; start of sequence

next: lodsd
      and eax,eax
      jz fim            ;; jump if end of sequence reached
      xor edx,edx
      div n
      and edx,edx
      jnz  next
mlt:  inc ecx           ;; found one more!
      jmp next

fim:  mov eax,ecx
      ret
multn endp
```