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

1. Uma sequência de elementos do tipo `dword` contém 364 pares de valores que representam as temperaturas mínima e máxima registadas em cada dia das 52 semanas do ano. O 1º elemento de cada par representa a temperatura mínima e o 2º a temperatura máxima.

[20] (a) Escreva a rotina `TMS` que, recorrendo a aritmética de inteiros, retorna a média das temperaturas máximas dos 7 dias de uma semana, arredondada ao inteiro mais próximo. O protótipo da rotina é: `TMS proto apt: ptr dword`.

Resposta:

```
TMS proto uses esi edx apt: ptr dword
    mov esi, apt     ; primeiro dia da semana
    mov ecx,7        ; sete dias
    xor eax,eax      ; soma
@@: add eax, dword ptr[esi+4]    ; adiciona temp maxima
    add esi,8                    ; temp maxima seguinte
    loop @b
    xor edx,edx      ; prepara divisao
    mov ecx,7
    div ecx          ; media em EAX
    cmp edx,4        ; se resto maior ou igual a 4
    jb @f            ; passa para o inteiro seguinte
    inc eax
@@: ret
TMS endp
```

[20] (b) Complete o segmento de código do programa apresentado que, recorrendo à rotina `TMS`, imprime o número da semana mais quente do ano e a média das suas temperaturas máximas.

```
include mpcp.inc
TMS proto start: ptr dword
.data
temps dword 11,15,10,16,09,17,08,17,09,18,10,18,10,18,
            11,16,10,17,09,18,08,18,09,18,10,18,10,18,...
msg byte "Semana mais quente: %d, temperatura: %d.",10,13,0
.code
main proc c
      mov esi,offset temps
      ;; ... código a completar
      invoke ExitProcess,0
main endp
```

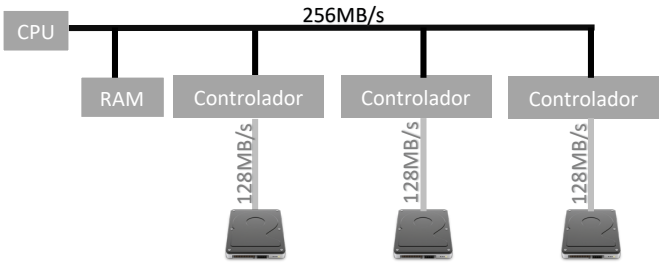
Resposta:

```
main proc c
      mov esi,offset temps
```

```
      xor ebx,ebx
      mov edi,1

next: invoke TMS,esi     ; calcula TMS
      cmp ebx,eax        ; novo maximo?
      jae @f             ; nao - passa a frente
      mov ebx,eax        ; atualiza maximo
      mov edx,edi        ; atualiza numero da semana
@@:   cmp edi,52         ; fim do ano?
      jae stop
      inc edi            ; proxima semana
      add esi,56         ; 14 dwords (7 pares) mais a frente
      jmp next
stop: invoke printf, offset msg, edx, ebx
      invoke _getch
      invoke ExitProcess,0
main endp
end
```

[30] 2. 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.]

Resposta:

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:

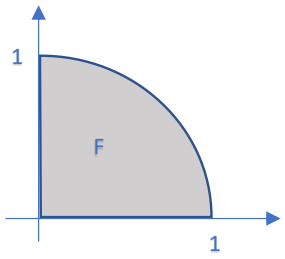
$$3\,\text{ms} + \frac{256\,\text{kB}}{128\,\text{MB/s}} = 5\,\text{ms}$$

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.

[20] 3. Na figura seguinte está representada uma região F à qual pertencem todos os pontos (x, y) tais que $x >= 0$, $y >= 0$ e $x^2 + y^2 <= 1$.



Implementar a sub-rotina `VerificaPonto` que recebe as coordenadas x e y de um ponto, determina se o mesmo pertence ou não à região F e apresenta no monitor a mensagem correspondente.

O protótipo da sub-rotina é:

```
VerificaPonto PROTO argx:real8, argy:real8, msgSIM:ptr byte, msgNAO:ptr byte
```

Resposta:

```
VerificaPonto PROC argx:real8, argy:real8, msgSIM:ptr byte, msgNAO:ptr byte
        mov ecx, 1
        cvtsi2sd xmm3, ecx
        xorps xmm4, xmm4
        movsd xmm1, argx
        movsd xmm2, argy
        comisd xmm1, xmm4
        jb fora
        comisd xmm2, xmm4
        jb fora
        mulsd xmm1, xmm1
        mulsd xmm2, xmm2
        addsd xmm1, xmm2
        comisd xmm1, xmm3
        ja fora
        invoke printf, msgSIM
        jmp fim
```

```
fora:
        invoke printf, msgNAO
fim:
        ret
VerificaPonto  ENDP
```

4. Apresenta-se a seguir uma parte do código gerado pelo *assembler* para a invocação de uma sub-rotina.

```
...
call SUBROT
add  esp, 4
```

A instrução `add` ocupa o endereço de memória `00B83EE0h`. A sub-rotina tem uma variável local do tipo `DWORD`.

O estado da pilha após a execução do prólogo de `SUBROT` é apresentado na tabela.

Endereço (hex.)	Conteúdo (hex.)
0059F10C	...
0059F108	80000000
0059F104	00B83EE0
0059F100	009B3A7E
0059F0FC	140011C0
0059F0F8	00000010

[10] (a) Explique o objetivo da instrução `"add esp, 4"` neste contexto.

Resposta:

A presença da instrução `"add esp, 4"` logo após a chamada de `SUBROT` revela que esta sub-rotina utiliza a convenção de chamada `C`, segundo a qual a responsabilidade de libertar o espaço ocupado pelos argumentos da sub-rotina na pilha é da responsabilidade do invocador. Ao somar `4` a `ESP` está-se a eliminar um elemento da pilha (a pilha decresce com endereços crescentes), garantindo assim que a pilha é deixada no estado em que se encontrava antes da invocação.

[10] (b) Indique o que representa cada item do conteúdo.

Resposta:

Após a sub-rotina invocada terminar é executada a instrução seguinte, ou seja, `"add esp, 4"`. Esta ocupa o endereço de memória `00B83EE0h`, significando que é este o endereço de retorno da sub-rotina, encontrando-se no endereço `0059F104` da pilha. Na posição anterior (endereço `0059F108`) está contido o único argumento da sub-rotina (conclusão decorrente da análise feita na alínea anterior).

Os conteúdos indicados nas três últimas linhas da tabela referem-se ao valor de `EBP`, conteúdo da variável local e valor de um registo a preservar.

Endereço (hex.)	Conteúdo (hex.)	Significado
0059F10C	...	(topo da pilha antes da invocação)
0059F108	80000000	Argumento da sub-rotina
0059F104	00B83EE0	Endereço de retorno da sub-rotina
0059F100	009B3A7E	Valor de `EBP`
0059F0FC	140011C0	Variável local
0059F0F8	00000010	Registo a preservar

[20] 5. A norma de um vetor $\vec{v} = (c_1, c_2, ..., c_n)$ é definida por $\sqrt{\sum_{i=1}^{n} c_i^2}$.

Pretende-se implementar uma sub-rotina `normvec` que calcula a norma utilizando dados empacotados. Assuma que a dimensão do vetor é múltipla de 4.

Considere o protótipo: `normvec PROTO ptV: ptr real4, n: dword, ptVN: ptr real4`.

O endereço da sequência de componentes do vetor é dado por `ptV`, a dimensão do vetor é `n` e o resultado é guardado no endereço de memória `ptVN`.

Resposta:

```
normvec PROC uses esi ptV: ptr real4, n: dword, ptVN: ptr real4
    mov    esi, ptV
    xorpd  xmm0, xmm0
    mov    ecx, n
    shr    ecx, 2          ; Nº iterações = n/4
@@: movups xmm1, [esi]     ; Lê 4 componentes do vetor
    mulps  xmm1, xmm1      ; Calcula o quadrado de cada componente
    addps  xmm0, xmm1      ; Acumula os 4 quadrados em 4 SPFP
    add    esi, 16         ; Endereço das próximas 4 componentes
    loop   @B
    haddps xmm0, xmm0
    haddps xmm0, xmm0    ; Os 4 SPFP em xmm0 têm a soma dos quadrados
    sqrtps xmm0, xmm0    ; Os 4 SPFP em xmm0 têm o valor da norma do vetor
    mov    esi, ptVN
    movss  real4 ptr [esi], xmm0    ; Guarda valor da norma em memória
    ret
normvec 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 as seguintes declarações de duas rotina iguais:

```
ROT1 PROC C APT:PTR BYTE, LEN:DWORD

ROT2 PROC   APT:PTR BYTE, LEN:DWORD
```

A propósito do processo de tradução para código máquina, qual é a afirmação verdadeira.

A. Os epílogos de `ROT1` e `ROT2` são iguais, mas os prólogos diferentes.

B. `ROT1` e `ROT2` têm o mesmo epílogo e prólogo, mas diferente código de chamada.

C. `ROT1` e `ROT2` têm o mesmo epílogo, prólogo, e código de chamada.

D. Os prólogos de `ROT1` e `ROT2` são iguais, mas os epílogos diferentes.

[5] (b) As variáveis locais de uma sub-rotina são criadas:

A. no segmento de dados, ficando acessíveis a outras sub-rotinas;

B. na pilha, não ficando acessíveis a outras sub-rotinas;

C. no segmento de dados, não ficando acessíveis a outras sub-rotinas;

D. na pilha, ficando acessíveis a outras sub-rotinas.

[5] (c) Considere as variáveis `x:real4` e `y:real8` às quais é atribuído o mesmo valor. Qual das afirmações é verdadeira:

A. Ocupam um número diferente de bytes em memória;

B. Ocupam o mesmo número de bytes em memória, mas y tem maior precisão do que x.

C. Ocupam o mesmo número de bytes em memória, mas y tem menor precisão do que x.

D. A metade mais significativa de y vale zero.

[5] (d) Considere o seguinte fragmento de código:

```
.data
val    REAL4     3.1415
.code
    insertps    xmm1,val,00101000b
    unpckhps    xmm1,xmm1
```

O valor final do registo XMM1 é:

A.

3.1415	3.1415	3.1415	3.1415

B.

3.1415	0	3.1415	0

C.

0	**0**	**3.1415**	**3.1415**

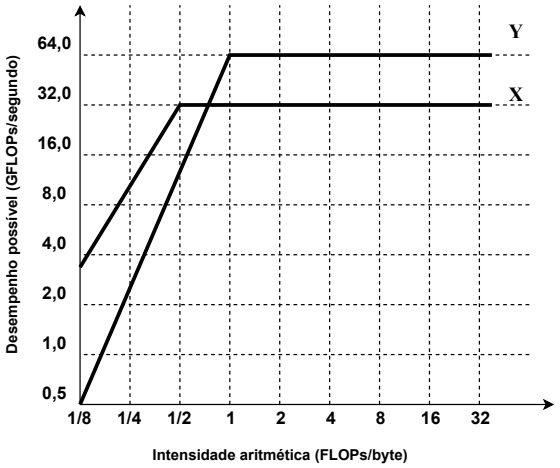
D.

0	3.1415	0	3.1415

[5] (e) Um multiprocessador de memória partilhada em que alguns acessos a memória são mais rápidos que outros (dependendo do processador e da posição de memória acedida) tem acessos a memória do tipo:

A. NUMA B. UMA C. sincronizados D. especulativos

[5] (f) O diagrama seguinte mostra as curvas de desempenho para dois computadores, X e Y.



Ambos os computadores podem executar os seguintes programas:
P1: intensidade aritmética 1/2, 800 GFLOPS
P2: intensidade aritmética 3, 2400 GFLOPS

Qual par apresenta o menor tempo de execução?

A. (P1, Y) B. (P2, X) C. (P2, Y) **D. (P1, X)**

7. Considerar o seguinte programa composto por dois ficheiros.

```
#include <iostream>
using namespace std;
extern "C" bool func(char *arg1, char *arg2, int n);
int main()
{
   char op1[]="479", op2[]="015", op3[]="74598", op4[]="28237";
   bool res;

   cout << op1 << ' ' << op2 << " --> ";
```

```
    res = func(op1, op2, 3);
    cout << op1 << ' ' << (res ? "(!!)" : "(OK)") << endl;
    cout << op3 << ' ' << op4 << " --> ";
    res = func(op3, op4, 5);
    cout << op3 << ' ' << (res ? "(!!)" : "(OK)") << endl;
    return 0;
}
```

```
1  include mpcp.inc
2
3  .code
4
5  func PROC C uses ebx edi esi \
6       arg1:ptr byte, arg2:ptr byte,
7       n:sdword
8
9         xor  al, al
10        xor  bl, bl
11        mov  ecx, n
12        mov  edi, arg1
13        mov  esi, arg2
14
15        add  esi, ecx
16        add  edi, ecx
17        dec  esi
18        dec  edi
19
20
21
22
23 ciclo: mov  dl, [edi]
24        mov  dh, [esi]
25        sub  dl, '0'
26        sub  dh, '0'
27        add  dl, dh
28        add  dl, bl
29        .IF dl >= 10
30           sub dl, 10
31           mov bl, 1
32        .ELSE
33           mov bl,0
34        .ENDIF
35        add  dl, '0'
36        mov  [edi], dl
37        dec  edi
38        dec  esi
39        loop ciclo
40
41        mov  al, bl
42 fim:   ret
43 func ENDP
44 end
```

Quando executado, o programa apresenta no monitor a seguinte mensagem:

```
479 015 --> 494 (OK)
74598 28237 --> 02835 (!!)
```

Nota: Justificar todas as respostas.

[10] (a) Quantas vezes é executado o corpo do ciclo das linhas 23–39 na *primeira* chamada de `func`?

> **Resposta:** A repetição do ciclo é controlada pela instrução `loop` da linha 39, que usa o valor de `ECX` para determinar se deve ser executada uma nova iteração. No corpo do ciclo não existe nenhuma instrução de salto para fora do ciclo, nem instruções que alterem o valor do registo `ECX`. Assim, valor de `ECX` antes do ciclo vai determinar o número de iterações. Este valor é definido na linha 11 como sendo igual a `n`. Portanto, o corpo do ciclo é repetido 3 vezes.

[10] (b) Que valores ficam no registo `DL` após cada execução da linha 28 no decurso da *primeira* chamada da sub-rotina?

Resposta: O valor de `DL` resulta da soma dos valores decimais dos dígitos (0–9) de cada posição correspondente das duas cadeias de carateres acrescentada do transporte da operação anterior (valor do registo `BL`, que pode ser apenas 0 ou 1). Os valores decimais são calculados nas linhas 26 e 26, a soma dos dígitos e do transporte é feita nas linhas 27–28. Assim, como os dígitos são processados da direita para a esquerda, temos sucessivamente: 9+5+0=14, 7+1+1=9 e 4+0+0=4.

[10] (c) Quantas vezes são executadas as instruções das linhas 30–31 na *segunda* chamada da sub-rotina e qual é a sua finalidade (para o algoritmo implementado pela sub-rotina)?

Resposta: A função procede à adição dígito a dígito, da direita para a esquerda, de dois números representados em ASCII. As linhas referidas determinam o transporte para a posição seguinte: se a soma dos dígitos e do transporte anterior for maior que 10 (a soma é necessariamente menor ou igual a 19), o transporte é 1 e o dígito dessa posição é o que fica quando se subtrai 10 ao valor em consideração (porque incrementar a próxima posição de 1 é equivalente a somar 10 à posição atual). [Nota: estas são apenas as regras usuais da adição de dois números decimais.]

Para os números dados, o transporte é diferente de 0 para: 8+7+0, 9+3+1, 4+8 e 7+2+1. Ao todo, as linhas referidas são executadas 4 vezes.

[10] (d) Qual é o significado do valor retornado pela sub-rotina `func`?

Resposta: Como de trata de uma sub-rotina que deve retornar um valor booleano, o resultado é dado pelo valor de registo `AL`. Neste caso, esse registo assume o último valor de `BL`, que representa o transporte a partir da posição mais significativa (mais à esquerda). Portanto, `AL=1` se o valor do transporte é 1, i.e., a soma não é exprimível com o número de dígitos disponíveis (*overflow*); neste caso, a zona de memória apontada por `arg1` (que é alterada pela sub-rotina para guardar a representação ASCII da soma dos valores) não fica com o conteúdo correto. Se `AL=0`, então a soma pôde ser calculada corretamente. Resumindo, o resultado 0 indica que a soma foi calculada corretamente, o resultado 1 indica que a soma não pode ser representada corretamente com a memória disponível (perde-se o dígito mais significativo da soma).