Atenção: Este exame tem 7 questões, num total de 200 pontos.

Análise de código

1. Considere o seguinte programa:

```
1  include \masm32\include\masm32rt.inc
2
3  func1 proto V: ptr sdword, dim: dword, num: dword
4  func2 proto valor: sdword, elem: dword
5
6  .data
7  seq sdword 3, -17, 7, 10, -5, 0, -1, -7
8  k   dword  7
9
10 .code
11 start:
12         invoke func1, offset seq, lengthof seq, k
13         print ustr$(eax), 13, 10
14         inkey
15         exit
16
17 func1 proc uses esi V: ptr sdword, dim: dword, num: dword
18         xor  edx, edx
19         mov  esi, V
20         mov  ecx, dim
21         jecxz sai
22   @@:   invoke func2, [esi], num
23         add  edx, eax
24         add  esi, 4
25         loop @B
26   sai:  mov  eax, edx
27         ret
28 func1 endp
29
30 func2 proc valor: sdword, elem: dword
31         mov  eax, valor
32         cmp  eax, 0
33         jns  numP
34         neg  eax
35   numP: cmp  eax, elem
36         mov  eax, 0
37         rcl  eax, 1 ; usa CF
38         ret
39 func2 endp
40 end start
```

[10] (a) Descreva a sub-rotina `func2` e indique a sua funcionalidade.

Resposta: A sub-rotina começa por verificar se o inteiro `valor` é negativo. Caso seja, calcula o seu simétrico. O valor absoluto assim determinado é depois comparado com o inteiro positivo `elem`. Sendo inferior, a sub-rotina devolve o valor 1 em EAX ou 0 no caso contrário. Portanto, a funcionalidade de `func2` é verificar se o valor absoluto de um inteiro (`valor`) é inferior a um determinado valor (`elem`).

[10] (b) Determine o valor apresentado no monitor após a execução do programa. Justifique.

Resposta: A sub-rotina `func1` calcula o número de elementos de uma sequência com valor absoluto inferior a um determinado limite. Conclui-se portanto que o valor apresentado é 4, pois há 4 elementos com valor absoluto inferior a 7.

[5] (c) Indique o número de vezes que a sub-rotina `func2` é executada. Justifique.

Resposta: `func2` é executada 8 vezes, correspondendo ao número de elementos da sequência processada.

[10] (d) Assuma que `ESP = 18FF88h` antes de executar pela primeira vez a instrução da linha 22. Apresente as alterações da pilha até ser executada pela primeira vez a instrução da linha 33, indicando para cada endereço o respectivo conteúdo.

Resposta:

endereço (hex.)	conteúdo (hex.)
18FF84	00000007
18FF80	00000003
18FF7C	end. retorno
18FF78	EBP anterior

[5] (e) Apresente o epílogo da sub-rotina `func1`.

Resposta:

```
pop  esi
leave
ret  12
```

Sistemas de entrada/saída

[Nota: Para simplificar os cálculos assuma k=10^3, M=10^6 e G=10^9.]

2. Considere um computador com dois discos:

- Um disco magnético tradicional que roda a 3000 RPM, tem uma latência média de busca de 30 ms, uma latência de controlador de 10 ms e transfere dados a 20 MB/s.
- Um disco SSD que usando memória *flash* consegue ler e escrever blocos de 4 kB a uma taxa de 10 MB/s, com uma latência de acesso desprezável.

Pretende-se projectar um controlador inteligente para alternar entre os discos consoante for mais eficaz. Justifique as suas respostas com cálculos.

[5] (a) Considerando uma transferência de 40 kB contíguos, qual é o tempo total de acesso para o disco SSD?

Resposta:

$$\frac{40\,\text{kB}}{10\,\text{MB/s}} = 4\,\text{ms}$$

[10] (b) Qual é o tempo de acesso da mesma transferência de 40 kB contíguos no caso do disco magnético? Considere que após começar a transferência, eventuais mudanças de pista demoram um tempo negligenciável.

Resposta: T_rotação $= 0.5 * 60000/3000 = 10\,\text{ms}$
T_acesso $= 30 + 10 + 10 + 40\,\text{kB}/20\,\text{MB} = 52\,\text{ms}$

[10] (c) Calcule o tamanho da transferência que o controlador inteligente deve usar como valor limite para decidir trocar entre o disco SSD e o disco magnético (i.e., o limiar de decisão).

Resposta: Temos de encontrar o tamanho de tranferência t para o qual os dois discos são equivalentes:
$0{,}05\,\text{s} + t/20\,\text{MB} = t/10\,\text{MB} \equiv$
$0{,}05\,\text{s} = t/10\,\text{MB} - t/20\,\text{MB} \equiv$
$0{,}05 = t \times 10\,\text{MB}/200\,\text{MB} \equiv$
$t = 200 \times 0{,}05/10 = 1\,\text{MB}$
Controlador inteligente: $t < 1\,\text{MB}$: Disco SSD; $t > 1\,\text{MB}$: Disco magnético.

3. Considere um computador com um CPU de 2 GHz e uma câmara de vídeo ligada por um barramento Firewire capaz de transferir blocos de 16 kB a 800 MB/s. Pretende-se usar a estratégia de *polling* para transferir dados da câmara para memória.

[5] (a) Considere um cenário em que o CPU tem de gastar 99% do seu tempo em outras tarefas que não *polling*. Sabendo que neste sistema, uma operação de *polling* consome 200 ciclos de relógio, qual o número máximo de operações de *polling* que podem ser realizadas num segundo pelo CPU?

Resposta: $2 \times 10^9 \times 0.01/200 = 1 \times 10^5$

[6] (b) Considerando apenas as restrições impostas pelo barramento Firewire, qual é o número máximo de blocos de 16 kB que podem ser lidos da câmara de vídeo num segundo?

Resposta: T_operação = $16\,\text{kB}/800\,\text{MB} = 2 \times 10^{-5} s$
N_operações = $\dfrac{1}{2 \times 10^{-5}} = 5 \times 10^4$

[4] (c) Com base nos cálculos anteriores, quantas operações de *polling* devem ser realizadas (por segundo) de forma a evitar perdas de dados e minimizar a ocupação de CPU?

Resposta: 50000 operações. Apesar do CPU suportar mais operações, estas não são necessárias, pois a câmara de video não iria tirar partido das mesmas.

Escolha múltipla

4. Para as alíneas seguintes, indique a única resposta correcta.

[10] (a) Uma das instruções seguintes dá erro ao ser compilada. Identifique-a.

A. `cmp ebx, 0ffh`

B. `mov edx, word ptr [edi]`

C. `rol ax, 15`

D. `rol eax, 33`

Resposta: B.

[10] (b) Considere o seguinte fragmento de código *assembly*:

```
mov al, 0AAh
mov bl, al
rol bl, 1
adc al, bl
```

Após a execução deste fragmento o conteúdo de `AL` é:

A. 0 B. 1 C. 255 D. 256

Resposta: A.

[10] (c) A utilização de `__asm` numa função escrita em C/C++ permite:

A. juntar à função o código máquina de outra função;

B. chamar uma sub-rotina em *assembly*;

C. incluir um bloco de instruções *assembly*;

D. assinalar que essa função não deve ser compilada.

Resposta: C.

[10] (d) Identifique a afirmação falsa sobre sistemas RAID:

A. RAID 5 é tão eficiente quanto RAID 0 ao nível de operações de escrita pequenas (1 bloco).

B. RAID 5 consegue sobreviver a falhas de múltiplos discos, desde que estes pertençam a grupos de protecção diferentes.

C. RAID 0 e RAID 1 são igualmente eficientes ao nível de operações de leitura.

D. RAID 5 é mais eficiente que RAID 4 ao nível de operações de escrita.

Resposta: A.

Programação

5. Uma palavra é considerada um palíndromo se puder ser lida da mesma forma da esquerda para a direita e da direita para a esquerda. Exemplos de palíndromos incluem "o", "ana", "osso"e "salas".

[20] (a) Escreva uma sub-rotina para testar se uma palavra é ou não um palíndromo. A rotina deve retornar 1 se a palavra for um palíndromo e 0 em caso contrário. Pode assumir que a palavra está escrita em minúsculas e que a mesma é composta por pelo menos uma letra.

```
palindromo PROTO PAL:PTR BYTE, N: DWORD
```

`PAL` é um apontador para a palavra a analisar e `N` o comprimento da mesma.

Resposta:

```
palindromo PROC PAL:PTR BYTE, N: DWORD
        mov ecx, PAL ; apontador início
        mov edx, ecx
        add edx, N ; apontador fim
        dec edx
@@:
        cmp ecx, edx
        jae _pal
        xor eax, eax
        mov al, byte ptr [ecx]
        mov ah, byte ptr [edx]
        inc ecx
        dec edx
        cmp al, ah
        je @B
        mov eax, 0
        jmp _ret
_pal:
        mov eax, 1
_ret:
        ret
palindromo endp
```

[20] (b) Considere agora um vector `vect` de caracteres (BYTEs) cujo conteúdo são palavras separadas por um e um só espaço (carácter 32 em ASCII). Use a sub-rotina da alínea anterior para escrever um fragmento de código que coloque em EAX o número de palavras presentes em vect que são palíndromos. Para simplificar, assuma que existe pelo menos uma palavra na frase e que esta é terminada por um espaço. Por exemplo, se `vect = "a ana ama saias mas odeia sugus "`, no final da execução EAX deverá conter o valor 5.

Resposta:

```
        mov esi, offset vect                    pop ecx
        mov ecx, lengthof vect                  add ebx, eax
        xor edi, edi ; tam. palavra             add esi, edi
        xor ebx, ebx ; contador                 mov edi, -1
_loop:                                  _next:
        mov al, byte ptr [esi]                  inc esi
        cmp al, 32                              inc edi
        jne _next                               loop _loop
        sub esi, edi                    _over:
        push ecx                                mov eax, ebx
        invoke palindromo, esi, edi
```

[20] 6. Apresente o código da rotina que calcula o produto interno de dois vectores de N números reais (N $>$ 0) usando a unidade de vírgula flutuante.

Sejam $X = [x_1, x_2, ..., x_N]$ e $Y = [y_1, y_2, ..., y_N]$. Então, o produto interno é dado por

$$X \cdot Y = x_1 \times y_1 + x_2 \times y_2 + ... + x_N \times y_N$$

O protótipo da rotina é:

```
prodint PROTO vectX:ptr real8, vectY:ptr real8, N: dword
```

Resposta: (Uma solução de entre várias possíveis.)

```
prodint PROC USES esi edi vectX:ptr REAL8, vectY:ptr REAL8, N:dword
    mov ecx, N
    mov esi, vectX
    mov edi, vectY
    fldz
@@: fld     real8 ptr [esi]
    fmul    real8 ptr [edi]
    fadd
    add     edi, 8
    add     esi, 8
    loop    @B
    ret
prodint ENDP
```

[20] 7. Considere a função $f(x), x \in \mathbb{R}$, definida por

$$f(x) = \begin{cases} x \sin(x + \pi) & \text{se} \quad x \geq 0 \\ \dfrac{1}{\sqrt{4 - x}} & \text{se} \quad x < 0 \end{cases}$$

Implemente a sub-rotina `rotF` que calcula a função $f(x)$ para qualquer valor de x usando a unidade de vírgula flutuante. O respectivo protótipo é:

```
rotF PROTO argX: REAL8
```

Resposta: (Uma solução de entre várias possíveis.)

```
.data
const4 REAL8 4.0

.code
rotF PROC argX:REAL8
    fld argX
    ftst
    fstsw ax
    sahf
    jb negativo
    fldpi
    fadd
    fsin
    fmul argX
    jmp fim
negativo:
    fld const4
    fsubr
    fsqrt
    fld1
    fdivr
fim:
    ret
rotF ENDP
```

Fim do enunciado.