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. Para $n \geq 0$ inteiro, o número de Fibonacci $F(n)$ define-se por:

$$F(n) = \begin{cases} 0 & \text{para } n = 0 \\ 1 & \text{para } n = 1 \\ F(n-1) + F(n-2) & \text{para } n \geq 2 \end{cases}$$

O seguinte programa compara duas sub-rotinas para calcular números de Fibonacci, `fib1` e `fib2` (assumir que $0 \leq n \leq 99$).

```
include mpcp.inc
     .data
tab         DWORD 100 dup(0)
cont1       DWORD 0
cont2       DWORD 0
fmt1        BYTE "fib1(%d)=%d  cont1=%d", 13, 10, 0
fmt2        BYTE "fib2(%d)=%d  cont2=%d", 13, 10, 0
fib1 PROTO C  N:DWORD
fib2 PROTO C  N:DWORD
     .code
fib1 PROC C N:DWORD
       inc    cont1
       mov    eax, N
       .IF eax >= 2
         dec    eax
         push   eax
         invoke fib1, eax
         pop    edx
         push   eax
         dec    edx
         invoke fib1, edx
         pop    edx
         add    eax, edx
       .ENDIF
       ret
fib1 ENDP


fib2 PROC C N:DWORD
       inc    cont2
       mov    eax, N
       .IF eax >= 2
         mov  ecx, tab[4*eax]
         cmp  ecx, 0
         je   @F
         mov  eax, ecx
         jmp  fim
@@:      dec  eax
         push eax
         invoke fib2, eax
         pop  edx
         push eax
         dec  edx
         invoke fib2, edx
         pop  edx
         add  eax, edx
         mov  ecx, N
         mov  tab[4*ecx], eax
       .ENDIF
fim:   ret
fib2 ENDP

main PROC C
   invoke fib1, 5
   invoke printf, offset fmt1, 5, eax, cont1
   invoke fib2, 5
   invoke printf, offset fmt2, 5, eax, cont2
   mov    cont2, 0
   invoke fib2, 5
   invoke printf, offset fmt2, 5, eax, cont2
   invoke ExitProcess, 0
main ENDP

END
```

O programa apresenta no monitor:

```
fib1(5)=5  cont1=15
fib2(5)=5  cont2=9
fib2(5)=5  cont2=1
```

[15] (a) Explicar o funcionamento de `fib1` e o valor de `cont1` (1ª linha apresentada no monitor).

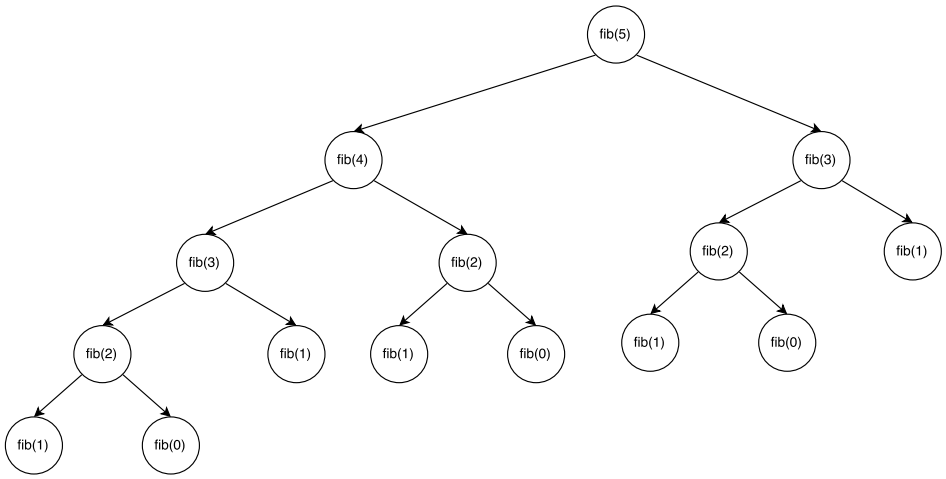
Resposta: A sub-rotina `fib1` implementa diretamente a definição, incrementando a variável global `cont1` a cada invocação. De acordo com a convenção `C`, o resultado estará no registo `eax`.

Para `n=0` e `n=1`, `fib1` retorna logo esses valores como resultados. Os outros caso são tratados nas linhas 14–24. A invocação da linha 17 calcula o valor de $F(n-1)$. Esse valor é guardado na pilha pela instrução seguinte.

A invocação da linha 21 calcula o valor de $F(n-2)$. Notar que o valor $n-2$ está guardado no registo `edx`: o valor de $n-1$ (do registo `eax)` é colocado na pilha na linha 16 e passado para `edx` na linha 18.

O valor de $F(n-1)$ é recuperado da pilha e colocado em `edx` (linha 22) e adicionado a $F(n-2)$ na linha seguinte.

Da definição, $F(5) = 5$, o que explica que a 1ª linha apresente `fib1(5)=5`. O valor do contador pode ser calculado desenhando a árvore das invocações de `fib1` e contando os nós (15).

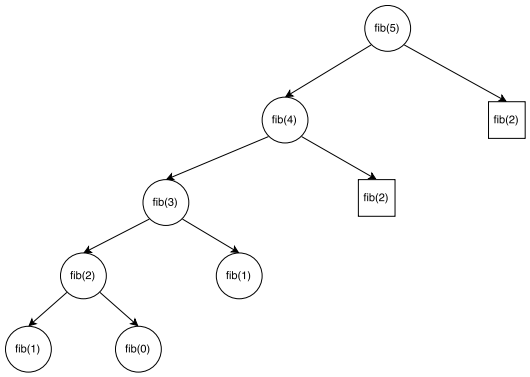


[15] (b) Explicar as diferenças entre `fib1` e `fib2`, bem como os valores de `cont2` (2ª e 3ª linhas apresentadas no monitor). Notar que a sub-rotina `fib2` usa a sequência global `tab`.

Resposta: A sub-rotina `fib2` incrementa a variável global `cont2` por cada invocação. A implementação é semelhante à da sub-rotina `fib1`, mas para o cálculo de $F(n), n > 2$, `fib2` consulta primeiro a posição `n` da tabela global `tab`, cujo conteúdo está inicialmente a zero. A consulta é realizada na linha 33. Se o conteúdo for zero, significa que o valor de $F(n)$ ainda não está na tabela é deve ser calculado como em `fib1`, mas chamando recursivamente `fib2`. Após calculado, o valor é guardado na tabela `tab`, posição índice `n` (código da linha 48).

Esta abordagem limita as invocações recursivas, já que estas só são efetuadas se o valor de $F()$ desejado ainda não tiver sido calculado. O primeiro cálculo de `fib2(5)` resulta na árvore de invocações seguinte (ao comparar com a anterior,

reparar que as subárvores `fib2(4)`, `fib2(3)` e `fib2(2)` apenas surgem uma vez (os nós quadrados representam valores obtidos da tabela). No total, a árvore tem 9 nós. A segunda invocação de `fib2(5)` encontra logo o valor na tabela, pelo que só é necessária uma invocação.



[10] (c) Apresente e justifique o conteúdo das 10 primeiras posições de `tab` no fim da execução do programa.

Resposta: A sub-rotina `fib2` e e respetivas chamadas recursivas levam ao armazenamento de todos os valores de $F(N)$ para $2 \leq n \leq 5$ na tabela. As posições de índice 0 ou 1 nunca são alteradas. Assim, o conteúdo pedido é:

`tab={0, 0, 1, 2, 3, 5, 0, 0, 0, 0}`.

2. Considere o seguinte programa. Antes da invocação da sub-rotina `fun` os valores de `ESP` e `EBP` são, respetivamente, `00B3FCCCh` e `00B3FD10h`. O endereço da instrução `push ebx` é `010E7C44h`.

```
fun PROC uses ebx V:DWORD, N:BYTE
  mov   ebx, V      ; (*)
  inc   ebx
  movzx eax, N
  xor   eax, V
  ret
fun ENDP
```

```
main proc c
  xor  ebx, ebx
  dec  ebx
  invoke fun, ebx, 10
  push ebx
  push eax  ; (**)
main endp
```

[10] (a) Apresente uma tabela com o estado da pilha resultante da execução do programa até à instrução assinalada com (*). Indique endereços e conteúdos em formato hexadecimal.

Resposta:

Endereço (hex)	Conteúdo (hex)	Obs.
00B3FCC8	0000000A	2º argumento da invocação de `fun`
00B3FCC4	FFFFFFFF	1º argumento da invocação de `fun`
00B3FCC0	010E7C44	endereço de retorno a `main`
00B3FCBC	00B3FD10	`EBP` anterior
00B3FCB8	FFFFFFFF	preservação de `EBX`

[10] (b) Repita a alínea anterior considerando agora a execução até à instrução assinalada com (**) inclusive.

Resposta: Ao terminar a execução da sub-rotina `fun` a pilha é deixada no estado em que estava antes da invocação de `fun`, pelo que o endereço do topo da pilha (dado por `ESP`) é `00B3FCCCh`.

Endereço (hex)	Conteúdo (hex)
`00B3FCC8`	`FFFFFFFF`
`00B3FCC4`	`FFFFFFF5`

[20] 3. Um sistema informático, possui um processador capaz de executar 2G ciclos/s, e dois discos com uma taxa de transmissão de 100 MB/s cada. Uma transferência por DMA tem um custo de início de 1800 ciclos e de finalização de 200 ciclos. Assumindo que os discos têm continuamente informação a enviar ao CPU (pior cenário), determine qual é o tamanho mínimo da transferência de DMA por forma a que o custo total de acesso aos discos seja, no máximo, 1% do tempo de CPU. Considere kB = 10^3 B, MB = 10^6 B.

Resposta:

Seja D o tamanho da transferência de DMA para um disco:

$$\text{Tempo de uma transferência de DMA: } T = \frac{D}{100 \times 10^6} = \frac{D}{10^8}$$

$$\text{Ciclos necessários por transferência: } N = 1800 + 200 = 2000 = 2 \times 10^3$$

$$\text{Ciclos necessários por unidade de tempo: } C = \frac{N}{T} = 2 \times 10^3 \times \frac{10^8}{D} = \frac{2 \times 10^{11}}{D}$$

$$\text{Percentagem do tempo de CPU: } \frac{C}{2 \times 10^9} = 0,01; \frac{2 \times 10^{11}}{2 \times 10^9} = 0,01 \times D; D = 10^4$$

O tamanho mínimo de uma transferência de DMA será pois de 10kB para um disco. Havendo dois discos iguais será proporcionalmente reduzida, ou seja, 5kB.

4. Considere 2 discos com as seguintes características:

- Um disco SSD capaz de ler e escrever blocos de 8kB a uma taxa de 40MB/s com uma latência de acesso desprezável;
- Um disco magnético tradicional que roda a 6000 RPM possuindo uma latência média de busca de 20ms, uma latência de controlador de 10ms e uma velocidade de transferência de dados de 80MB/s;

[10] (a) Qual dos discos anteriores apresenta um tempo de transferência inferior no caso de um ficheiro de 800 MB? (Para o disco magnético assuma que o tempo gasto em eventuais mudanças de pista é desprezável).

Resposta: Disco SSD:

$$\frac{800\text{MB}}{40\text{MB/s}} = 20\text{s}$$

Disco Magnético:

$$0,02 + 0,01 + 0,5 \times \frac{60}{6000} + \frac{800\text{MB}}{80\text{MB/s}} = 0,03 + 0,005 + 10 = 10,035\text{s}$$

O disco Magnético apresenta um tempo de transferência inferior.

[10] (b) Admitindo que a taxa de transferência dos dois discos é duplicada, em qual dos discos o aumento de desempenho é superior? Justifique devidamente a resposta.

Resposta:

Disco SSD:

Nova taxa de transferência: 80MB/s

Novo tempo:

$$\frac{800\text{MB}}{80\text{MB/s}} = 10\text{s}$$

$$\frac{10}{20} = 0,5 = 50\,\%$$

Disco Magnético:

Nova taxa de transferência: 160MB/s

Novo tempo:

$$0,02 + 0,01 + 0,5 \times \frac{60}{6000} + \frac{800\text{MB}}{160\text{MB/s}} = 0,03 + 0,005 + 5 = 5,035\text{s}$$

$$\frac{5,035}{10,035} = 0,502 = 50,2\,\%$$

Como $50\,\% < 50,2\,\%$ conclui-se que o aumento de desempenho é superior no disco SSD.

[30] 5. Considere a função $y = \dfrac{1}{\sqrt{|\sin x|}}$. Assuma que $x \neq k\pi$ (k inteiro).

Implemente a sub-rotina `roty` com o protótipo

```
roty PROTO x:REAL4, ypt:PTR REAL4
```

que calcula o valor da função em `x` e guarda o resultado no endereço indicado por `ypt`. Para o cálculo de $\sin x$ assuma que tem disponível a sub-rotina `seno` com o protótipo `seno PROTO x:REAL4, spt:PTR REAL4` sendo `spt` o endereço onde é devolvido o resultado.

Resposta:

```
roty PROC x:REAL4, ypt:PTR REAL4
  mov    eax, ypt
  invoke seno, x, eax
  movss  xmm0, real4 ptr [eax]   ; sen(x)
  mulss  xmm0, xmm0              ; sen^2(x)
  sqrtss xmm0, xmm0              ; |sen(x)|
  rsqrtss xmm0, xmm0             ; 1/raiz(|sen(x)|)
  movss  real4 ptr [eax], xmm0
  ret
roty 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 dois sistemas RAID (*Redundant Arrays of Inexpensive Disks*) destinados a armazenar 60 TB, sem contar com qualquer redundância. O sistema S1 usa tecnologia RAID 1 e o sistema S2 usa tecnologia RAID 4 com 6 discos num grupo de proteção. A capacidade de armazenamento de cada um dos sistemas é:

A. S1: 120 TB; S2: 70 TB B. S1: 60 TB; S2: 72 TB
C. S1: 120 TB; S2: 80 TB D. S1: 120 TB; S2: 72 TB

[5] (b) Indique o conteúdo da *flag* de *carry* (CF) e da *flag* de *overflow* (OF) após execução do seguinte fragmento de código:

```
mov eax, 0FFFFF080h
mov ah, 80h
add al, ah
```

A. CF=0, OF=0 B. CF=0, OF=1 **C. CF=1, OF=1** D. CF=1, OF=0

[5] (c) Considere dois acessos independentes a um mesmo disco magnético. Num, o tempo de acesso a 10 blocos consecutivos é 10,1 ms. No outro, o tempo de acesso a 30 blocos consecutivos é 10,3 ms. A soma dos tempos de busca, rotação e do controlador é:

A. 10,09 ms **B. 10 ms** C. 9,99 ms D. 1,01 ms

[5] (d) Considerar a sub-rotina que começa por:

```
rotina PROC C uses ebx SEQ:PTR WORD, VAL:DWORD
LOCAL  tmp[2]:SDWORD
```

Quantos bytes ocupa a moldura de uma invocação desta sub-rotina?

A. 32 B. 20 C. 24 **D. 28**

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

```
     .data
valores  BYTE  3,8,11,3,6,9,15
var      DWORD 200
   .code
```

```
mov ecx, lengthof valores
mov edi, offset valores
mov al, 11
repne scasb
```

No final da execução do código indicado, o valor do registo `EDI` é:

A. `5` **B.** `offset valores + 3` C. `lengthof valores - 4` D. `offset var`

[5] (f) Quantos bytes de memória são ocupados pelas declarações:

```
pi   REAL8 3.1415
seq  REAL4 3.0, 9.99
```

A. 3 B. 24 **C. 16** D. 12

Nome (legível): _____

7. Um contador regista em memória, de minuto a minuto, a potência consumida por um sistema de iluminação. Sabe-se que uma das lâmpadas está permanentemente ligada e não há mais do que uma lâmpada a ligar/desligar no mesmo minuto. Por exemplo, a sequência {35, 35, 5, 5, 65, 95, 35} mostra que uma lâmpada de 5 W está permanentemente ligada, uma lâmpada de 30 W é ligada no minuto 0 e desligada no minuto 2, uma lâmpada de 60 W é ligada no minuto 4 e desligada no minuto 6, e uma lâmpada de 30 W é ligada no minuto 5 e mantém-se ligada até ao fim do registo.

[10] (a) Escreva a sub-rotina `resid` que determina o consumo residual (o consumo da lâmpada que está sempre ligada), desconta-o a cada elemento da sequência e retorna-o. No exemplo anterior a sub-rotina altera a sequência para {30, 30, 0, 0, 60, 90, 30} e retorna o valor 5. Protótipo: `resid proto seq:ptr sdword, n:dword`, onde `seq` aponta o início da sequência e `n` é o número de elementos.

Resposta:
Basta calcular o valor mínimo da sequência, numa segunda fase subtraí-lo a cada elemento e por fim retorná-lo.

```
resid proc uses esi seq:ptr sdword,
               n: dword
     mov esi, seq
     mov ecx, n
     mov eax, 07fffffffh
@@:  cmp eax, [esi]
     jl cont
     mov eax, [esi] ;;Mínimo em EAX
cont: add esi,4
     loop @b
     mov esi, seq
     mov ecx, n
@@:  sub [esi], eax ;;Desconta-o
     add esi, 4
     loop @b
     ret ;; Retorna-o em EAX
resid endp
```

[10] (b) Escreva a sub-rotina `event` que identifica variações de consumo, pela diferença entre cada elemento e o anterior, a partir do segundo elemento. No exemplo, a sub-rotina altera a sequência para {30, 0, -30, 0, 60, 30, -60}. Protótipo: `event proto seq:ptr sdword, n:dword`, onde `seq` aponta o início da sequência e `n` é o número de elementos.

Resposta:
Basta calcular as referidas diferenças começando pelo último elemento da sequência (para não estragar os dados) e parar no segundo elemento.

```
event proc uses esi seq:ptr sdword,
                      n: dword
     mov ecx, n
     dec ecx
     mov esi,ecx
     shl esi,2
     add esi,seq
@@:  mov eax, [esi]
     sub eax, [esi-4]
     mov [esi], eax
     sub esi,4
     loop @b
     ret
event endp
```

[20] (c) Complete o programa principal que, recorrendo às sub-rotinas desenvolvidas, processa a sequência de consumos e indica qual o consumo residual e quais os minutos em que se ligam/desligam as lâmpadas. Para o exemplo anterior, deverá imprimir:

```
Lampada de 5W sempre acesa
------------------------------
Minuto 0: lampada de 30W acende
Minuto 2: lampada de 30W apaga
Minuto 4: lampada de 60W acende
Minuto 5: lampada de 30W acende
Minuto 6: lampada de 60W apaga
```

```
include mpcp.inc
resid proto seq: ptr sdword, n: dword
event proto seq: ptr sdword, n: dword
.data
    consumos sdword 35, 35, 5, 5, 65, 95, 35
    txt1 byte "Lampada de %dW sempre acesa",10,13
    txt2 byte "--------------------------------",10,13,0
    txt3 byte "Minuto %d: lampada de %dW ",0
    txt4 byte "acende",10,13,0
    txt5 byte "apaga",10,13,0
  .code
main proc c
local minuto:dword
         mov esi, offset consumos
         mov ebx, lengthof consumos
         invoke resid, esi, ebx
         invoke printf, offset txt1, eax
         invoke printf, offset txt2
         invoke event, esi, ebx

         mov minuto,0
@@:      mov eax,[esi]
         cmp eax,0
         jz next
         jg acesa
apagada: neg eax
         invoke printf, offset txt3, minuto, eax
         invoke printf, offset txt5
         jmp next
acesa:   invoke printf, offset txt3, minuto, eax
         invoke printf, offset txt4
next:    add esi,4
         inc minuto
         cmp minuto,ebx
         jnz @b

         invoke _getch
         invoke ExitProcess, 0
main endp
end
```