Nº de ordem: _____

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação
Microprocessadores e Computadores Pessoais
Teste 1

1º ano
2015-04-08
Duração 1:30
Sem consulta

Nome: _____ **Nº de estudante:** _____

Atenção: Este teste tem 12 questões em 4 páginas, num total de 200 pontos.

Parte I — Questões de Escolha Múltipla

Cada questão tem uma resposta certa. Respostas erradas não descontam.
As respostas às questões de escolha múltipla devem ser assinaladas com × na grelha seguinte.
Apenas as respostas indicadas na grelha são consideradas para efeitos de avaliação.

	Questão													
Opção	1	2	3	4	5	6	7	8	9	10	11a	11b	11c	11d
A		×		×				×	×					
B	×									×		×	×	
C			×			×								×
D					×		×				×			

Pontos: _____ / 140

[10] 1. Quantos bytes de memória são reservados pela declaração `var SWORD 5 dup(0)`?

A. 5 **B. 10** C. 20 D. 40

[10] 2. Qual das seguintes instruções dá erro ao compilar?

A. `inc [edi]` B. `mov ax, [esi]` C. `movsx eax, dx` D. `imul eax`

[10] 3. Qual é o valor de `ax` após a execução do seguinte código:

```
mov  ax, 324AH
stc
rcr  ax, 2
```

A. `4A32H` B. `0C92H` **C.** `4C92H` D. `8C92H`

[10] 4. Assuma a declaração `v sword -8, 4, 7, 9`. Após execução de `mov cl, type v`, o valor de `cl` é:

A. 2 B. 4 C. 8 D. 16

[10] 5. Assuma a seguinte declaração:

```
.data
val1 sword ?
val2 sbyte ?
```

Qual dos seguintes fragmentos de código permite calcular o valor de `val1` × `val2`?

A.
```
mov   ax, val1
imul  val2
```

B.
```
movzx eax, val2
imul  val1
```

C.
```
imul  val1, val2
```

D.
```
movsx ax, val2
imul  val1
```

[10] 6. Qual dos seguintes fragmentos de código troca os valores dos registos `eax` e `ebx`?

A.
```
push eax
push ebx
pop  ebx
pop  eax
```

B.
```
mov  eax, ebx
mov  ebx, eax
```

C.
```
xor  eax, ebx
xor  ebx, eax
xor  eax, ebx
```

D.
```
push eax
mov  eax, ebx
pop  eax
```

[10] 7. Considerar o seguinte fragmento:

```
.data
seq BYTE     3, 5, 1, 11, 10, 33, 4
.code
      mov    esi,OFFSET seq
      mov    ecx,LENGTHOF seq
prox: test   BYTE PTR [esi], 1
      pushfd
      inc    esi
      popfd
      loopnz prox
```

Quantas instruções são executadas?

A. 22 B. 37 C. 35 **D. 27**

[10] 8. Considerar o seguinte fragmento de código.

```
.data
valores  SWORD 30000, 2000, 5000, -50, -50
.code
     mov    ecx, LENGTHOF valores
     mov    edi, OFFSET valores
     xor    ax, ax
     xor    bx, bx
L1:  add    bx, [edi]
     cmovo  bx, ax
     add    edi, 2
     loop   L1
```

Qual é o valor final do registo `bx`?

A. -100 B. 32767 C. 0 D. 36950

[10] 9. Qual dos seguintes elementos não pode ser encontrado num registo de ativação de uma sub-rotina?

A. variável global B. variável local C. endereço de retorno D. argumento da sub-rotina

[10] 10. A sub-rotina `rotx` (convenção `stdcall`) tem o seguinte protótipo: `rotx PROTO v1: DWORD, v2:BYTE`.

Durante a invocação com "`invoke rotx, ecx, 20`" quantos bytes da pilha são ocupados pelos argumentos da sub-rotina?

A. 5 **B. 8** C. 12 D. 9

11. Considere o seguinte programa:

```
 1  include mpcp.inc
 2
 3          .data
 4  k       dword  7
 5  SEQ     sdword 2, -4, 0, 9, 1, -7, -8
 6  msg     byte   "Resultado: %d",0
 7
 8          .code
 9  main:   mov    edx, k
10          mov    ebx, k
11          neg    ebx
12          xor    eax, eax
13          mov    esi, offset SEQ
14          mov    ecx, lengthof SEQ
15  L:      .if    (sdword ptr [esi] > ebx) && (sdword ptr [esi] < edx)
16            inc eax
17          .else
18            .if sdword ptr [esi] >= edx
19              mov [esi], edx
20            .else
21              mov [esi], ebx
22            .endif
23          .endif
24          add    esi, type SEQ
25          loop   L
26          invoke printf, offset msg, eax
27          invoke _getch
28          invoke ExitProcess, 0
29  end main
```

[10] (a) Após execução, o programa imprime o valor:
A. 5 B. 3 C. 2 **D. 4**

[10] (b) A sequência SEQ após a execução do programa é:
A. 2, -4, 0, -7, 1, 7, 7 **B. 2, -4, 0, 7, 1, -7, -7**
C. 7, -7, 7, 7, 7, -7, -7 D. 2, -4, 0, 9, 1, -7, -8

[10] (c) O número de vezes que a instrução na linha 21 é executada é
A. 4 **B. 2** C. 3 D. 1

[10] (d) O código entre as linhas 18 e 22, inclusive, é equivalente a:

A.
```
        cmp sdword ptr [esi],edx
        jge @C1
        mov [esi], ebx
  @C1:  mov [esi], edx
```

B.
```
        cmp sdword ptr [esi],edx
        jb  @C3
        mov [esi], edx
        jmp @C4
  @C3:  mov [esi], ebx
  @C4:
```

C.
```
        cmp sdword ptr [esi],edx
        jl  @C1
        mov [esi], edx
        jmp @C2
  @C1:  mov [esi], ebx
  @C2:
```

D.
```
        cmp sdword ptr [esi],edx
        jl  @C1
        mov [esi], ebx
        jmp @C2
  @C1:  mov [esi], edx
  @C2:
```

Parte II — Exercício de programação

Atenção: Responder no enunciado.

[60] 12. Complete o programa apresentado abaixo que imprime quantos múltiplos de `val` existem em `seq`. O programa deve funcionar para qualquer sequência de DWORDs. Garanta que o resultado é escrito na seguinte forma (exemplo para os dados declarados):

```
A sequencia tem 3 multiplos de 5.
Adeusinho.
```

```
include mpcp.inc

       .data
val    DWORD    5
seq    DWORD    3, 40, 15, 8, 1, 13, 5, 17, 21
fmt    BYTE    "A sequencia  tem %d multiplos de %d.",10,13,"Adeusinho.",0

        .code
main:   mov     esi, offset seq      ; apontar para primeiro elemento
        mov     ecx, lengthof seq    ; número de elementos
        xor     ebx,ebx              ; contador de múltiplos
@@:     xor     edx, edx             ; parte mais significativa do dividendo a 0
        mov     eax, [esi]           ; parte menos significativa do dividendo
        div     val
        cmp     edx,0                ; verificar se resto é zero
        jne     notzer
        inc     ebx                  ; encontrado um múltiplo
notzer: add     esi, type seq        ; próximo elemento da sequência
        loop @b                      ; repetir

        invoke printf, offset fmt,  ebx, val
        invoke _getch
        invoke ExitProcess,0
end main
```