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
2017-03-16
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.

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

Pontos: _____ / 140

[10] 1. Indique o conteúdo de `ecx` após execução do seguinte fragmento de código:

```
mov ecx, 00FF00AAh
shr ecx, 4
rcr ecx, 4
adc ecx, 0h
```

A. `FE000001h` **B.** `5000FF01h` C. `A000FF00h` D. `5000FF00h`

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

```
mov ebx, 0FFFFFA0Ah
sar ebx, 4
add bl, bh
```

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

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

A. `movzx esi, byte ptr [ecx]` B. `neg cf` C. `mul ax, bx` D. `mov [edi], -1`

[10] 4. Indique o conteúdo de `AH` após execução do seguinte fragmento de código:

```
xor eax, eax
mov ah, -1
rcr ah, 2
rcl ax, 1
or  ah, al
```

A. `FFh` B. `FEh` C. `7Eh` D. `01h`

[Nota: nenhuma resposta está correta; atribuída cotação total.]

[10] 5. Indique o fragmento de código que calcula o valor de (v1^2 + 1)/v2. Assuma que v1 e v2, assim como os resultados intermédios, são valores sem signal com 16 bits e que inicialmente AX=v1.

A.
```
mul ax
add ax, 1
div v2
```

B.
```
mul eax
inc eax
sub dx, dx
div v2
```

C.
```
mul ax
inc ax
cwd
div v2
```

D.
```
mul v1
inc ax
cwde
div v2
```

[10] 6. Qual das afirmações é verdadeira relativamente ao seguinte fragmento de código?

```
cmp dx, 7fffh
jg  fim
```

A. O salto nunca é tomado.

B. O salto é tomado se dx $\geq$ 8000h.

C. O salto só não é tomado se dx = 7fffh.

D. O salto é tomado se edx > 7fffffffh.

[10] 7. Considere uma sub-rotina com o seguinte protótipo:

```
teste proto arg1:ptr byte, arg2:ptr sdword, arg3:sword
```

Qual das opções de invocação é válida?

A. `invoke teste, ah, esi, cx`

B. `invoke teste, offset seq, [bx], sizeof seq`

C. `invoke teste, dword ptr [edi], offset seq, offset V2`

D. `invoke teste, offset V1, esi, -1`

[10] 8. Em qual dos fragmentos de código resulta al = min(bl,cl), considerando operandos com sinal?

A.
```
mov al,bl
cmp al,cl
jae fim
mov al,cl
fim:
```

B.
```
mov al,bl
cmp al,cl
jg fim
mov al,cl
fim:
```

C.
```
mov al,bl
cmp al,cl
jle fim
mov al,cl
fim:
```

D.
```
mov al,bl
cmp al,cl
jb fim
mov al,cl
fim:
```

[10] 9. Após execução do seguinte fragmento de código, qual das afirmações é verdadeira:

```
cmp   eax, 127
cmovg eax, 127
cmp   eax, -128
cmovl eax, -128
```

A. EAX será 127 ou −128.

B. EAX será −1.

C. EAX estará entre −128 **e** 127**.**

D. EAX será 0.

[10] 10. Inicialmente ESP=45002248h, indique o valor final de ESP após a execução do fragmento:

```
push  edx
push  ax
pushd 100
```

A. 45002252h **B.** 4500223Eh C. 4500223Ch D. 45002254h

11. O seguinte programa valida o identificador ISBN de um livro.

    O ISBN $x_1\,x_2\,x_3-x_4-x_5\,x_6-x_7\,x_8\,x_9\,x_{10}\,x_{11}x_{12}-x_{13}$ (x_i é um algarismo decimal) é válido se o valor da expressão seguinte for múltiplo de dez:

$$x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12} + x_{13}$$

```
 1      include mpcp.inc
 2          .data
 3  dados byte '978-3-16-148410-0'
 4  dez   dword 10
 5  msg1  byte 'ISBN valido',13, 10, 0
 6  msg2  byte 'ISBN invalido', 13, 10, 0
 7          .code
 8  main PROC C
 9          mov ecx, lengthof dados
10          mov edi, offset dados
11          xor eax, eax
12          xor dl, dl
13  ciclo:  mov bl, [edi]
14          .IF bl >= '0' && bl <= '9'
15              sub    bl, '0'
16              movzx  ebx, bl
17              add    eax, ebx
18              .IF dl != 0
19                  add eax, ebx
20                  add eax, ebx
21              .ENDIF
22              not dl
23          .ENDIF
24          inc  edi
25          loop ciclo
26          xor  edx, edx
27          div  dez
28          .IF edx == 0
29             mov edi, offset msg1
30          .ELSE
31             mov edi, offset msg2
32          .ENDIF
33          invoke printf, edi
34          invoke _getch
35          invoke ExitProcess, 0
36  main endp
37  end
```

[10] (a) Para os dados do programa, quantas vezes é executado o ciclo que começa na linha 13?
A. 18 B. 13 **C. 17** D. 10

[10] (b) Para os dados do programa, quantas vezes é executado o bloco de instruções das linhas 19–20?
A. 6 B. 13 C. 8 D. 17

[10] (c) O teste da linha 28 tem como objetivo determinar se:
A. a sequência `dados` termina com zero; B. não ocorreu uma exceção na divisão;
C. o valor de `eax` é divisível por 10; D. a sequência `dados` tem pelo menos 10 dígitos.

[10] (d) Qual das seguintes alterações de código pode alterar o resultado do programa?
A. linha 12: `mov dl,al` **B. linha 33:** `invoke printf,[edi]`
C. linha 25: `loopnz ciclo` D. linha 16: `movsx ebx, bl`

Parte II — Exercício de programação

Atenção: Responder no enunciado.

[60] 12. Completar o programa que determina quantos pontos do plano estão (estritamente) à direita e abaixo do ponto (Px, Py). As coordenadas dos pontos a classificar estão guardadas numa sequência (chamada `pontos`) com cada par de coordenadas em posições sucessivas (abcissa seguida de ordenada). Assumir que a sequência tem pelo menos um ponto. O programa deve funcionar corretamente para qualquer número de pontos. Não usar diretivas `.IF/.ENDIF`.

Exemplo: Para os dados indicados abaixo, o primeiro ponto da sequência tem coordenadas (-2,0) e o programa deve escrever:
`Numero de pontos abaixo e ‘a direita de (Px,Py):  2`

```
include mpcp.inc
        .data
pontos  sword  -2, 0, 2, -2, 3, 0, 0, 0, 0, -3, 1, 0
Px      sword 1
Py      sword 1
msg     byte 'Numero de pontos abaixo e ‘a direita de (Px,Py): %d',13, 10, 0
        .code
main PROC C
        mov ecx, lengthof pontos
        shr ecx, 1
        mov esi, offset pontos
        xor edx, edx

ciclo:
        mov ax, [esi]
        mov bx, [esi+2]
        cmp ax, Px
        jle L1
        cmp bx, Py
        jge L1
        inc edx
L1:     add  esi, 4
        loop ciclo

        invoke printf, offset msg, edx

        invoke _getch
        invoke ExitProcess, 0
main ENDP
END
```