NOME: **Nº:**

Notas:

1. Duração estimada para a prova: **40 minutos**.
2. Para cada questão, numeradas de **1** a **5**, são apresentadas 2 hipóteses alternativas: responda a **apenas uma alínea**, a) ou b).
3. As respostas têm de ser convenientemente **justificadas**, incluindo o raciocínio ou os cálculos que efetuar.
4. **Não são permitidas**: (i) máquinas de calcular e (ii) notas auxiliares de memória.
5. A avaliação por questão obedece ao critério: não-satisfaz (**0**), satisfaz com erros (**0.8**), certa com falhas (**1.0**) e totalmente certa (**1.2**).

1.

a) Na norma ASCII, a sequência alfabética das minúsculas/maiúsculas "`a`" e "`A`" iniciam-se respetivamente pelos valores 141_8 e 101_8. Usando a notação octal **represente** a palavra "`JaDe`".

b) Na codificação BCD (*Binary-Coded Decimal*) cada dígito decimal é representado usando 4 bits de informação, por exemplo $9=1001_2=0x9$. **Represente** o valor "`2013`" em BCD usando a notação base 16.

2. Considere as funções C em baixo e os respetivos excertos de código de montagem após compilação. **Comente** as linhas terminadas pelo caráter "`;`" referindo sempre que possível o código C que lhe deu origem.

a)

```
int func (int arg, char r){
   switch (arg)
   { case 1:
       return r+1;
       break;
     default:
       return arg;
     break;}                    }
```

```
func:
      ...
   movl    8(%ebp), %eax      ; %eax<- int arg (1º arg)
   cmpl    $1, %eax           ;
   movb    12(%ebp), %dl      ;
   je      .L6                ;
.L1:
   movl    %ebp, %esp         ; coloca o %esp a apontar para o mesmo sítio que o frame pointer %ebp
   popl    %ebp               ;
   ret
.L6:
   movsbl  %dl,%eax           ; %al <- char r, e estende o resto de %eax com o bit do sinal
   incl    %eax               ;
   jmp      .L1
```

b)

```
int func (short arg, int r)  {
   for (; arg > 0; arg--)
       r-=arg;
  return r;                        }
```

```
func:
      ...
   movl    8(%ebp), %edx      ; %edx <- short arg (1º arg, colocado na stack com 32 bits)
   testw   %dx, %dx           ;
   movl    12(%ebp), %ecx     ;
   jle     .L8                ;
.L6:
   movswl  %dx,%eax           ; %ax <- short arg, e estende o resto de %eax com o bit do sinal
   decl    %edx               ;
   subl    %eax, %ecx         ; faz a operação r=r-arg
   testw   %dx, %dx           ;
   jg      .L6                ;
.L8:
   movl    %ecx, %eax         ; %eax<- r, o valor a devolver pela função
   movl    %ebp, %esp         ; coloca o %esp a apontar para o mesmo sítio que o frame pointer
   popl    %ebp               ;
   ret
```

3. Considere o processador IA-16, semelhante ao do IBM PC original (inteiros: 16-bits em complemento para 2).

a) Se nos registos `%ax` e `%cx` estiverem respetivamente os inteiros `0x94` e `0xc`, **indique**, em decimal, o resultado da operação expressa pela instrução `addw %cx,%ax`. **Apresente os cálculos!**

b) **Indique** se o banco de registos deste processador está preparado ou não para conter valores escalares inteiros pertencentes ao intervalo `[-9000 , +40000]` . **Justifique!**

4. Considere a representação de números em vírgula flutuante, baseada na norma IEEE, **com 12 bits**. O expoente é representado por **5 bits**, em excesso de $2^{(n-1)}$, e para a mantissa usam-se **6 bits**. Neste formato, o valor decimal de um nº normalizado é expresso por $V= (-1)^S * 1.F * 2^{(Exp-16)}$

a) **Represente** naquele formato, com arredondamento da mantissa, se necessário, o valor $-26375*10^{-3}$

b) **Indique** o valor (em decimal) representado pelo padrão binário "`1100 1111 1011`".

5. Imediatamente após a conclusão da execução duma instrução, considere a execução duma nova instrução do IA-32 (*little endian*), representada em *assembly* por: `movl %eax,-12(%edi,%eax,4)`.

A instrução em binário ocupa 4 células de memória: as 3 primeiras células contêm a sequência de valores `0x83`, `0x7d`, e `0xc5` correspondente ao código da operação seguido da especificação da localização dos operandos, e a 4ª célula contém a constante `-12`. Considere os seguintes conteúdos de registos:

%eax	0x210
%edi	0x8c20420
%esp	0x8c20444
%eip	0x800c322
%ebp	0x8c28f0c

a) **Apresente**, por ordem cronológica, toda a informação que circula apenas no barramento de dados (32-bits) na execução integral desta instrução.

Data Bus

tempo →

b) **Qual** o endereço da célula de memória, em **hexadecimal**, que irá conter o valor `0x02` resultante da execução desta instrução? **Justifique!**