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. Em BCD (*Binary-Coded Decimal*) cada dígito decimal é representado usando 4 bits de informação, por exemplo $9_{(10)} = 1001_{(2)} = 0x9$. A seguir sugerem-se duas hipóteses alternativas para reduzir o volume de informação.

a) Usar: **(i)** 3 bits, `000...101` para representar os dígitos `0...5`; **(ii)** a combinação `110` para indicar que os 2 bits que se seguem, `00...11`, codificam os dígitos `6...9`; **(iii)** a combinação `111` para significar a repetição do dígito anterior. Por exemplo, $294_{(10)}$ é codificado por `010 11011 100`.
Mostre como codificaria o valor **725883**$_{(10)}$.

b) Usar: **(i)** as sequências `0000` a `1001` para representar os dígitos `0...9` e **(ii)** as sequências `1010` a `1111` para codificar 2, 3, 4, 5, 6 e 7 repetições do dígito anterior .
Mostre como codificaria o valor **6 3 333 8 2 9 99999 7**.

2. Considere as funções abaixo e os respetivos excertos de código de montagem, após compilação. Para as linhas de C com asteriscos, **mostre e explique** o código de montagem correspondente.

a)

```
   int funcA (int val, char op){
   **  if (op == '+')      /* '+' ASCII 0x2b
   **    return val+2;
       else
   **     if (op=='*')
   **       return val*4;  /* '*' ASCII 0x2a
         return val;        }

   funcA:   ...
    1    movb    12(%ebp), %dl
    2    movl    8(%ebp), %eax
    3    cmpb    $43, %dl
    4    leal    2(%eax), %ecx
    5    je      .L1
    6    leal    0(,%eax,4), %ecx
    7    xorl    %eax, %eax
    8    cmpb    $42, %dl
    9    movl    %ecx, %eax
.L1:
   11    leave
   12    ret
```

b)

```
   int funcB (char * s, char d){
   ** int i=0;
   ** for (;s[i]!='0'; i++) /* '0'ASCII 0x30
   **    if (s[i] == d )
   **      return i;
      return 0;                 }

    funcB:
     1    movl    8(%ebp), %esi
     2    movb    (%esi), %al
     3    xorl    %edx, %edx
     4    cmpb    $48, %al
     5    movb    12(%ebp), %bl
     6    je      .L9
.L7:
     8    cmpb    %bl, %al
     9    movl    %edx, %ecx
    10    je      .L1
    11    incl    %edx
    12    movb    (%edx,%esi), %al
    13    cmpb    $48, %al
    14    jne     .L7
.L9:
    16    xorl    %ecx, %ecx
.L1:
    18    movl    %ecx, %eax
          ...
          ret
```

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

a) Considere que o compilador alocou o registo `%ax` para representar uma variável inteira, e que neste momento contém a dimensão de um bloco de memória: `24kB`. **Mostre em binário** o conteúdo de `%ax`.

b) **Apresente**, justificando, a gama de valores inteiros que é possível representar usando qualquer um dos registos `%ax`, `%bx`, `%cx`, ou `%dx`.

4. Considere a representação de números reais usando uma versão reduzida da norma IEEE 754 com 16 bits (6 bits para o expoente em excesso de $2^{(n-1)}-1$, 9 para a mantissa e 1 para o sinal; não esquecer os casos de exceção). O valor decimal de um nº normalizado representado com este formato vem dado por

$$V= (-1)^S * 1.F * 2^{(Exp-31)}$$

a) **Represente** naquele formato, com arredondamento da mantissa se necessário, o valor $-26375*10^{-2}$

b) **Apresente**, justificando, o intervalo de valores positivos que é possível representar no formato normalizado (em decimal).

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

A instrução em binário ocupa 4 células de memória. O 2º operando, em memória, é o valor `0x9e28`.

Considere os seguintes conteúdos de registos:

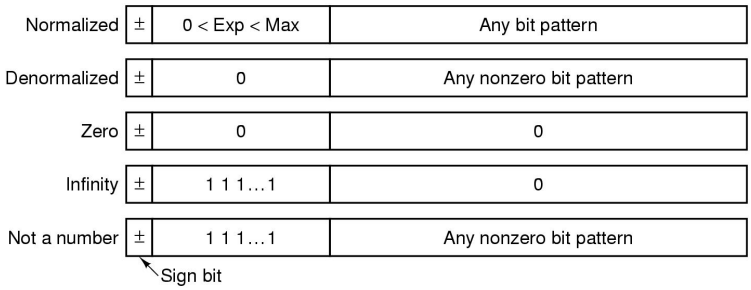
%eax	0x210
%edi	0x8c20420
%esp	0x8c20444
%eip	0x800c322
%ebp	0x8c28f0c

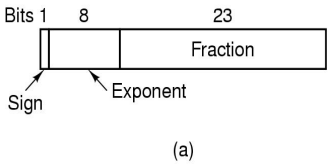
a) **Apresente**, por ordem cronológica, toda a informação que circula apenas no barramento de endereços (32-bits) na execução integral desta instrução.

Address Bus

b) O resultado desta adição, um valor de 32 bits, vai ser armazenado na memória a partir do endereço especificado na instrução. **Indique** em binário o valor que vai ser armazenado na 2ª célula de memória.

Notas de apoio (norma IEEE 754)





(a)

Valor decimal de um fp em binário:

precisão simples, normalizado: $\mathbf{V = (-1)^S * (1.F) * 2^{E-127}}$

precisão simples, desnormalizado: $\mathbf{V = (-1)^S * (0.F) * 2^{-126}}$