Nome:	**Nº:**

1. Responda no próprio enunciado, no espaço reservado para esse efeito. Apresente sempre **a justificação da solução**, incluindo os cálculos que efetuar (se precisar de mais espaço utilize o verso da folha).

2. Não são permitidas máquinas de calcular. É permitido o uso de uma folha A4, manuscrita, para consulta.

3. As 10 questões têm a **mesma cotação: 20 pontos cada**, para uma classificação total de 20 valores (duração 1h 30m).

1. Considere a seguinte função (incompleta!) em C e a compilação para *assembly* utilizando o `gcc`.

```
int xxx(int *vec)
{
    int i = 0, m = 0;
    ?? /* 1b_1 */
    {
        if (vec[i] ?? /* 1b_2 */ )
        {
           m = vec[i];
        }
    ?? } ?? /* 1b_1 */
    return(m);
}
```

```
xxx:    pushl   %ebp
        movl    %esp, %ebp
        pushl   %ebx
        xorl    %ecx, %ecx
        xorl    %edx, %edx
        movl    8(%ebp), %ebx
.L2:    movl    (%ebx,%edx,4), %eax
        cmpl    %eax, %ecx
        jge     .L3
        movl    %eax, %ecx
.L3:    incl    %edx
        cmpl    $49, %edx
        jle     .L2
        movl    %ecx, %eax
        popl    %ebx
        leave
        ret
```

a) **Indique,** justificando com as instruções *assembly* correspondentes, os registos atribuídos ao argumento `vec` e às variáveis locais `i` e `m`.

Variável	**Registo**	**Instrução(ões) *assembly***	**Comentários**
argumento `vec`			
variável local `i`			
variável local `m`			

b) Identifique as instruções *assembly* que implementam as duas estruturas de controlo do programa C e complete as partes em falta no programa (`1b_1` e `1b_2`).

Parte a completar	**Instruções *assembly***	**Código e justificação**
Estrutura `1b_1`		
Condição `1b_2` do `if`		

c) Assinale no programa *assembly* todas as instruções que leem dados da memória *(indicar na figura).*

d) O programa C incompleto tem dois acessos a cada elemento de `vec`. **Indique,** justificando de que tipo de localidade se trata (temporal ou espacial) e de que forma o compilador tirou partido dessa localidade.

e) **Apresente** uma estimativa do número máximo de instruções executadas em cada chamada à função `xxx`. Apresente os cálculos que efetuar.

Nome:	**Nº:**

2. Considere o estado de execução do programa anterior após uma interrupção (*breakpoint*) na execução com o `gdb`.

```
(gdb) disas xxx

0x080483a8:  pushl   %ebp
0x080483a9:  movl    %esp,%ebp
0x080483ab:  pushl   %ebx
0x080483ac:  xorl    %ecx,%ecx
0x080483ae:  xorl    %edx,%edx
0x080483b0:  movl    0x8(%ebp),%ebx
0x080483b3:  nop
0x080483b4:  movl    (%ebx,%edx,4),%eax
0x080483b7:  cmpl    %eax,%ecx
0x080483b9:  jge     0x80483bd
0x080483bb:  movl    %eax,%ecx
0x080483bd:  incl    %edx
0x080483be:  cmpl    $0x31,%edx
0x080483c1:  jle     0x80483b4
0x080483c3:  movl    %ecx,%eax
0x080483c5:  popl    %ebx
0x080483c6:  leave
0x080483c7:  ret
```

Registos: (parte *info registers*)

```
  %eax=0x00000006     %eip=0x080483b4
  %ebx=0xbfffaff0     %esp=0xbfffafd4
  %ecx=0x00000006     %ebp=0xbfffafd8
  %edx=0x00000002
```

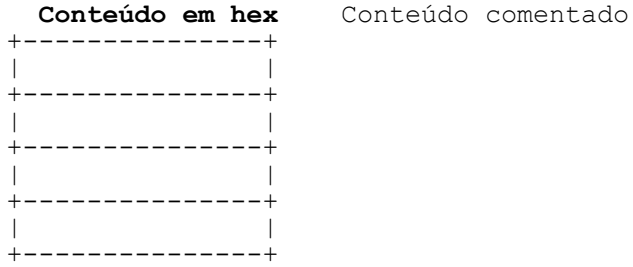
Memória (dados): (x/16 $esp)

```
0xbfffafd4:
  0xbfffaff0 0xbfffb0c8 0x080483ec 0xbfffaff0
0xbfffafe4:
  0x00003730 0x006797b9 0x00dee09c 0x00000003
0xbfffaff4:
  0x00000006 0x00000011 0x0000000f 0x0000000d
0xbfffb004:
  0x0000000f 0x00000006 0x0000000c 0x00000009
```

a) **Indique/calcule** o valor que seria armazenado em `%eax` após a execução de uma instrução `leal (%ebx,%edx,4),%eax`. Indique também uma possível expressão em C que pudesse originar esta instrução.

b) **Indique** em que endereço de memória foi colocado o ponto de paragem (*breakpoint*). Que registos serão alterados se a execução continuar até encontrar de novo esse ponto de paragem? Qual o seu novo valor?

c) **Complete** a figura seguinte com o conteúdo da *stack frame* associado à função `xxx`. Indique no lado esquerdo a posição apontada pelos registos `%esp` e `%ebp`. Inclua no lado direito de cada célula um comentário explicando e seu conteúdo (nota: cada célula representa um bloco de 32 bits, ou seja, 4 posições de memória).



d) **Indique** em que zona de memória foi alocado o vetor apontado pelo argumento `vec`, **justificando** com valores da figura. Indique também em que altura esse espaço será libertado.

3. Considere que a função é compilada para uma arquitetura RISC (e.g., ARM64), com 32 registos. **Indique** as alterações mais significativas ao *assembly* da função e o correspondente impacto no desempenho.