Nº: Nome:

Esta prova é constituída por 5 páginas. A resposta às perguntas deve ser feita nos espaços reservados no enunciado e deve incluir **sempre** a respetiva justificação. **Não é permitido** o uso de notas de consulta. A prova tem a duração de **1h30** (no teste modelo considere a duração **2h00**).

1. No fragmento de código de montagem abaixo, que **estrutura** de controlo está subjacente? Justifique a resposta a partir da escrita de uma hipótese de função em C, que deu origem a este código de montagem.

```
        ...
        movl    8(%ebp), %eax
        movl    12(%ebp), %ecx
        movl    %eax, %edx
        cmpl    %ecx, %eax
        jl      .L1
        movl    %ecx, %edx
.L1:
        movl    %edx, %eax
        leave
        ret
```

```
int funcao ( int a, ...

 {









}
```

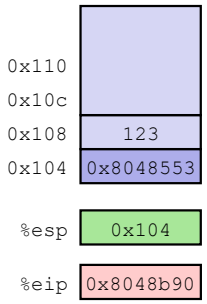
2. Considerando as declarações na figura abaixo: ***(i)*** represente em memória (posição, deslocamento) o argumento **<r>**; ***(ii)*** complete a função C a partir da interpretação do respetivo código compilado.

```
struct rec {
   int i;
   struct rec *n;
   int a[4];
};
```

```
int funcao
  (struct rec *r, int idx)
{
  return _____________;
}
```

```
...
movl    12(%ebp), %eax
movl    8(%ebp), %edx
movl    8(%edx,%eax,4), %eax
leave
```

3. A figura seguinte representa a memória e dois registos num ponto de paragem de um programa em execução através do **gbd**. A posição da memória **0x8048b90** contém o valor **C3**, que corresponde ao código da instrução **ret** do IA32.
   Com a ajuda desta figura**,** explique que alterações ocorrem no estado do programa (memória e/ou registos) quando se executa um comando **stepi** do **gdb** após a paragem.




4. Suponha que a próxima estrutura foi compilada numa máquina com Windows, onde cada tipo de dados primitivo de tamanho K bytes deve estar guardado num endereço múltiplo de K (alinhado de K).

```
struct {
   float  a;
   short  b;
   int   *c;
   char   d;
   double e[4];
} qt;
```

**(i)** Qual é o deslocamento de cada membro da estrutura **qt** em relação ao início desta estrutura?

**(ii)** Qual o número de bytes ocupado pela estrutura?

**(iii)** Reorganize os membros da estrutura de modo a minimizar o espaço desperdiçado.

O programa C, incluído na última folha, foi compilado e posteriormente executado, num ambiente equivalente ao do servidor das sessões laboratoriais, através do **gdb.** Na figura, o código desmontado representa **duas** versões da função **`seqParaInt`**, obtidas com o **gcc** e usando as opções de compilação **`-O0`** e **`-O2`**.
**Justifique todas as respostas!**

**5.** Considere os valores produzidos pela execução dos comandos do depurador **gdb**, nas condições de compilação **–O0**, visível na parte central da última folha.

a) ***(i)*** Qual o valor do registo %esp, imediatamente antes de executar a linha **`0x080483c7`** e ***(ii)*** qual o endereço da instrução, na função ***`seqParaInt`***, em que está suspensa a execução do programa?

b) Em que **endereços** estão localizadas as variáveis locais: <**i**> e <**sum**> e quais os respetivos **valores**, no ponto de paragem.

c) Em que endereço está localizado o parâmetro **`str`** da função e qual a sequência de carateres lidos pela função **`scanf("%s",str)`** no programa principal?

d) Com base na informação disponível, identifique na função **main** ***(i)*** o endereço de início do seu contexto na pilha (valor do registo %ebp) e ***(ii)*** o endereço de retorno da função ***`seqParaInt`***.

e) Identifique no código de montagem as linhas responsáveis pela execução da instrução **`i++`** do código C (linha 15).

6. Considere**,** na última folha, a listagem do código de montagem obtido com a opção de compilação **`-O2`**.

   a) Identifique/explique as otimizações existentes face à versão do código **–O0**, no que diz respeito: ***(i)*** à manipulação das variáveis locais e ***(ii)*** ao argumento da função ***`seqParaInt`***.

   b) Identifique/explique as linhas de código usadas para efetuar a instrução C, na linha 11, **sum=-1**.

   c) Apresente as razões que justificam a utilização dos pares de instruções: (**pushl %esi, popl %esi**), (**pushl %ebx, popl %ebx**)?

   d) Na linha **`<8048421:75 ??>`** substitua "**`??`**" pelo valor conveniente.

   e) Comente as seguintes linhas de código estabelecendo a relação com o código C:

```
80483f9:  movl   0x8(%ebp),%esi
80483fc:  movb   (%esi),%al
80483fe:  xorl   %ebx,%ebx
8048400:  xorl   %edx,%edx
8048402:  testb  %al,%al
8048404:  je     8048423 <+0x2f>
8048406:  movb   %al,%cl
```

**`gcc -Wall -O0 -g -o ProgSeqParaInt ProgSeqParaInt.c`**

```
0x080483c4 < +0>:     pushl  %ebp
0x080483c5 < +1>:     movl   %esp,%ebp
0x080483c7 < +3>:     subl   $0x8,%esp
0x080483ca < +6>:     movl   $0x0,-0x4(%ebp)
0x080483d1 < +13>:    movl   $0x0,-0x8(%ebp)
0x080483d8 < +20>:    movl   -0x4(%ebp),%eax
0x080483db < +23>:    addl   0x8(%ebp),%eax
0x080483de < +26>:    cmpb   $0x0,(%eax)
0x080483e1 < +29>:    jne    0x80483e5 < +33>
0x080483e3 < +31>:    jmp    0x804842c < +104>
0x080483e5 < +33>:    movl   -0x4(%ebp),%eax
0x080483e8 < +36>:    add    0x8(%ebp),%eax
0x080483eb < +39>:    cmpb   $0x2f,(%eax)
0x080483ee < +42>:    jle    0x80483fd < +57>
0x080483f0 < +44>:    movl   -0x4(%ebp),%eax
0x080483f3 < +47>:    addl   0x8(%ebp),%eax
0x080483f6 < +50>:    cmpb   $0x39,(%eax)
0x080483f9 < +53>:    jg     0x80483fd < +57>
0x080483fb < +55>:    jmp    0x8048406 < +66>
0x080483fd < +57>:    movl   $0xffffffff,-0x8(%ebp)
0x08048404 < +64>:    jmp    0x804842c < +104>
0x08048406 < +66>:    movl   -0x8(%ebp),%edx
0x08048409 < +69>:    movl   %edx,%eax
0x0804840b < +71>:    sall   $0x2,%eax
0x0804840e < +74>:    addl   %edx,%eax
0x08048410 < +76>:    leal   (%eax,%eax,1),%edx
0x08048413 < +79>:    movl   -0x4(%ebp),%eax
0x08048416 < +82>:    addl   0x8(%ebp),%eax
0x08048419 < +85>:    movsbl (%eax),%eax
0x0804841c < +88>:    leal   (%eax,%edx,1),%eax
0x0804841f < +91>:    subl   $0x30,%eax
0x08048422 < +94>:    movl   %eax,-0x8(%ebp)
0x08048425 < +97>:    leal   -0x4(%ebp),%eax
0x08048428 < +100>:   incl   (%eax)
0x0804842a < +102>:   jmp    0x80483d8 < +20>
0x0804842c < +104>:   movl   -0x8(%ebp),%eax
0x0804842f < +107>:   leave
0x08048430 < +108>:   ret
```

**Ficheiro ProgSeqParaInt.c**

```
/* Converte uma sequência de carateres para
   o inteiro positivo correspondente     */

 1 #include<stdio.h>
 2 #define TAM  8
 3 #define ZERO 48
 4 #define NOVE 57
 5
 6
 7 int seqParaInt(char str[]){
 8   int i=0,sum=0;
 9   while(str[i]!='\0'){
10     if(str[i]< ZERO || str[i] > NOVE){
11       sum= -1;
12       break;}
13     else{
14       sum = sum*10 + (str[i] - ZERO);
15       i++;}}
16   return sum;
17 }
18
19
20 int main(){
21   char str[TAM];
22   int valInt;
23
24   printf("Digite o numero positivo");
25   scanf("%s",str);
26
27   if ((valInt = seqParaInt(str)) >=0 )
28     printf("Valor: %d",valInt);
29   else
30     printf("Erro\n");
31   return 0;
32 }
```

**`(gdb) x/64 $ebp -12`**

```
0xbfffea0c: 0x00 0x00  0x00 0x00 0xfb 0x0f 0x00 0x00
0xbfffea14: 0x03 0x00  0x00 0x00 0x58 0xea 0xff 0xbf
0xbfffea1c: 0x71 0x84  0x04 0x08 0x40 0xea 0xff 0xbf
0xbfffea24: 0x40 0xea  0xff 0xbf 0x58 0xea 0xff 0xbf
0xbfffea2c: 0xd9 0x84  0x04 0x08 0x25 0xae 0x6a 0x00
0xbfffea34: 0xec 0xea  0xff 0xbf 0x58 0xea 0xff 0xbf
0xbfffea3c: 0xf4 0x3f  0x7d 0x00 0x34 0x30 0x39 0x31
0xbfffea44: 0x00 0x84  0x04 0x08 0x00 0x00 0x00 0x00
```

**`(gdb) info registers`**

```
eax            0xffb          4091
ecx            0x7d4420       8209440
edx            0xffa          4090
ebx            0x7d3ff4       8208372
esp            0xbfffea10     0xbfffea10
ebp            0xbfffea18     0xbfffea18
esi            0x573ca0       5717152
edi            0x0            0
eip            0x8048425      0x8048425
```

**`gcc -Wall -O2 -g -o ProgSeqParaInt ProgSeqParaInt.c`**

```
 80483f4:       55                      pushl  %ebp
 80483f5:       89 e5                   movl   %esp,%ebp
 80483f7:       56                      pushl  %esi
 80483f8:       53                      pushl  %ebx
 80483f9:       8b 75 08                movl   0x8(%ebp),%esi
 80483fc:       8a 06                   movb   (%esi),%al
 80483fe:       31 db                   xorl   %ebx,%ebx
 8048400:       31 d2                   xorl   %edx,%edx
 8048402:       84 c0                   testb  %al,%al
 8048404:       74 1d                   je     8048423 < +0x2f>
 8048406:       88 c1                   movb   %al,%cl
 8048408:       8d 41 d0                leal   0xffffffd0(%ecx),%eax
 804840b:       3c 09                   cmpl   $0x9,%al
 804840d:       77 1a                   ja     8048429 < +0x35>
 804840f:       0f be c1                movsbl %cl,%eax
 8048412:       43                      incl   %ebx
 8048413:       8d 14 92                leal   (%edx,%edx,4),%edx
 8048416:       8d 54 50 d0             leal   0xffffffd0(%eax,%edx,2),%edx
 804841a:       8a 04 33                movb   (%ebx,%esi,1),%al
 804841d:       84 c0                   testb  %al,%al
 804841f:       88 c1                   movb   %al,%cl
 8048421:       75 ??                   jne    8048408 < +0x14>
 8048423:       5b                      popl   %ebx
 8048424:       89 d0                   movl   %edx,%eax
 8048426:       5e                      popl   %esi
 8048427:       c9                      leave
 8048428:       c3                      ret
 8048429:       ba ff ff ff ff          movl   $0xffffffff,%edx
 804842e:       eb f3                   jmp    8048423 < +0x2f>
```