Ficha de Avaliação nº 1

(Ficha Tipo)

Programação em *assembly*

Número	Nome Completo

Instruções

- A resolução da ficha é **individual** e tem a duração de **45** minutos.
- Qualquer tentativa de fraude será sancionada com a invalidação de **todas** as fichas de avaliação prática do semestre.
- Todas as questões devem ser resolvidas neste enunciado.
- Pode consultar os conjuntos de instruções do IA32 e MIPS32 disponíveis na página web da disciplina.

A função apresentada na tabela 1 calcula, de forma recursiva, a soma de todos os elementos do `array` de inteiros `arr`, com `nelem` elementos, processando em cada invocação o elemento com índice `ind`.

Tabela 1 - Código em C

```
int soma (int *arr, int ind, int nelem)
{
      int ret=0;

      if (ind<nelem)
            ret = arr[ind] + soma(arr, ind+1, nelem);
      return (ret);
}
```

Na tabela 2 é apresentado o código *assembly* gerado pelo gcc sem qualquer nível de optimização.

Tabela 2 - Assembly IA32 sem optimização

```
soma:
      pushl %ebp
      movl  %esp, %ebp
      subl  $4, %esp            # BLOCO 1
      movl  $0, -4(%ebp)        # BLOCO 1
      movl  12(%ebp), %eax      # BLOCO 2
      cmpl  16(%ebp), %eax      # BLOCO 2
      jge   .L3                 # BLOCO 2
      subl  $4, %esp            # BLOCO 3
      pushl 16(%ebp)            # BLOCO 3
      movl  12(%ebp), %eax      # BLOCO 3
      incl  %eax                # BLOCO 3
      pushl %eax                # BLOCO 3
      pushl 8(%ebp)             # BLOCO 3
      call  soma                # BLOCO 3
      addl  $16, %esp           # BLOCO 3
      movl  %eax, %ecx
      movl  12(%ebp), %eax      # BLOCO 4
      imull $4, %eax, %edx      # BLOCO 4
      movl  8(%ebp), %eax       # BLOCO 4
      movl  (%eax,%edx), %eax   # BLOCO 4
      addl  %ecx, %eax
      movl  %eax, -4(%ebp)
.L3:
      movl  -4(%ebp), %eax      # BLOCO 5
      movl  %eax, %eax
      leave
      ret
```

Questão 1 – Identifique a funcionalidade de cada bloco de instruções etiquetado.

BLOCO 1 –
BLOCO 2 –
BLOCO 3 –

BLOCO 4 –
BLOCO 5 –

Na tabela 3 é apresentado o código *assembly* gerado pelo gcc com algumas optimizações.

Tabela 3 - Assembly IA32 com optimizações

```
soma:
      pushl %ebp
      movl  %esp, %ebp
      pushl %esi
      pushl %ebx
      movl  8(%ebp), %esi
      movl  12(%ebp), %ebx
      movl  16(%ebp), %edx
      movl  $0, %eax
      cmpl  %edx, %ebx
      jge   .L3
      subl  $4, %esp
      pushl %edx
      leal  1(%ebx), %eax
      pushl %eax
      pushl %esi
      call  soma
      addl  $16, %esp
      imull $4, %ebx, %ecx
      addl  (%esi,%ecx), %eax
.L3:
      leal  -8(%ebp), %esp
      popl  %ebx
      popl  %esi
      popl  %ebp
      ret
```

Questão 2 – Identifique e explique as optimizações feitas ao código anterior.

Questão 3 – Escreva o código em *assembly* do MIPS correspondente ao cálculo do endereço de `arr[ind]` e respectiva adição com o valor devolvido pela função `soma()` assumido que:

- O registo `$a0` contem o parâmetro `arr`;
- O registo `$a1` contem o parâmetro `ind`;
- O registo `$v0` contem o valor devolvido pela função `soma()`.