NOME: **Nº:**

1. Considere a figura anexa com um programa C desenvolvido e executado no servidor de apoio às sessões laboratoriais. A função `calcula(n)` começa por gerar um número TAM de valores aleatórios, entre **`0..n`** que armazena num vetor; depois calcula e devolve o valor médio dos elementos presentes no vetor.

2. As 11 questões nesta prova para 14 valores estão cotadas para **1 valor**, com exceção das questões **4**, **5** e **6** que valem **2 valores** cada.

1. Este pedaço de código em C foi compilado para *assembly* por 2 vezes: uma sem qualquer otimização, outra com a opção `-O2`. Analise as 2 versões em *assembly* e para cada uma dessas versões **mostre** como foram alocadas as 2 variáveis escalares locais e o argumento para a realização das operações (se a registos/memória, e quais/onde).

2. Identifique claramente, e justificando, a que parte do código C correspondem as seguintes instruções *assembly* da versão compilada sem otimização:

```
0x0804840f:  cmpl   $0x1f,-0xc(%ebp)
0x08048413:  jle    0x8048417
0x08048415:  jmp    0x8048438

0x08048431:  lea    -0xc(%ebp),%eax
0x08048434:  incl   (%eax)
```

3. Indique, justificando qual a função do comando do depurador apresentado na figura anexa [`(gdb) x /6xw $ebp-12`] e qual a utilidade deste comando na análise da execução deste programa.

4. Analise bem os dados fornecidos na figura do fim com o código e **indique, justificando (i)** qual a próxima instrução *assembly* a executar imediatamente após o *breakpoint* e **(ii)** qual o endereço da instrução `call` na `main`, que invocou a função `calcula(n)` (nota: a instrução `call` ocupa 5 *bytes*).

5. Apresente o quadro de ativação (*stack frame*) da função `calcula(n)`, na versão compilada com `-O2` **indicando** claramente todos os campos pertinentes e com a máxima informação possível.

6. Analisando de novo os dados fornecidos, **indique,** justificando **(i)** o conjunto de células de memória que foram reservadas para armazenar o vetor de valores aleatórios em `calcula(n)`, e **(ii)** o valor em decimal de `vector[2]` que foi criado na execução deste programa.

NOME: **Nº:**

---

7. Considere que a constante `TAM` foi definida como sendo 4, em vez de 32, e que tudo o resto nos dados fornecidos na figura do fim com o código estão corretos. **Indique,** justificando**,** o valor devolvido pela função.

8. **Substitua justificando** os '**??**' que aparecem no endereço `0x804845f` (listagem do código otimizado, desmontado com `objdump`) pelo valor correto deste *byte*.

9. Considere a representação de valores em vírgula flutuante conforme apresentada na PCE: 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), sendo o valor decimal de um nº normalizado dado por

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

Indique, justificando, em quantos dígitos decimais pode confiar num número com esta representação.

10. Considere a continuação da execução deste código da func `calcula(n)` (na versão *assembly* com `-O2`), após o *breakpoint*. Indique qual a 1ª instrução que irá ser executada que contenha um operação de escrita em memória, e qual o endereço de memória que vai ser especificado.

11. Considere também a execução numa arquitetura típica RISC das primeiras 6 instruções da função `calcula(n)` (na versão *assembly* com `-O0`). **Analise e compare** o código que seria gerado pelos 2 compiladores (IA-32 e RISC) em termos apenas de acessos à memória.

Use a sintaxe do IA-32 se quiser mostrar algum código para a arquitetura RISC.

Rascunho