

Execução de instruções e ISA-32

Teste 2

Nº	Nome
----	------

Nota: Apresente sempre o raciocínio ou os cálculos que efectuar; o não cumprimento desta regra equivale à não resolução do exercício.

1. Um colega seu usou um editor de texto para criar o ficheiro `funct2.c` com o código de uma função em C. Depois, usando o `gcc` para a arquitectura IA32, gerou sucessivamente os ficheiros `prog.s`, `prog.o`, e finalmente o ficheiro executável `prog`.
Caracterize sucintamente, cada um destes passos que conduziram à criação do ficheiro executável a partir do ficheiro com o código fonte em C.

2. Num PC com IA32, considere o seguinte estado do PC imediatamente antes de ser executada a instrução equivalente a `addl %ecx, -4(%ebp)`:
 - os registos `%ecx`, `%ebp` e `%eip` contêm, respectivamente, os seguintes valores em hexadecimal: `0x204`, `0x8f0480c`, `0x8c12004`;
 - cada célula de memória contém um valor que coincide com o *byte* menos significativo do seu endereço.

Apresente cronologicamente e em **hexadecimal**, toda a informação que irá circular no barramento de dados durante a execução integral da instrução acima. **Justifique** esses valores, considerando os valores nos **registos** e nas **células de memória** neste enunciado.

Nº	Nome
----	------

3. Considere os pressupostos do exercício anterior.

Indique todas as células de memória e todos os registos que foram modificados durante a execução integral daquela instrução, **explicando** sucintamente quais os novos valores que vão lá ser colocados.

4. **Identifique** no fragmento de código *assembly*, abaixo, qual a **estrutura de controlo** presente, e **mostre** as instruções em *assembly* que estão directamente relacionadas com a implementação daquela estrutura.

Notas:

1. A função recebe 2 argumentos: o 1º é um apontador para a variável `loc`, e está em memória à distância de **8 bytes** do local apontado pelo `%ebp`; o 2º argumento é a variável `n`, do tipo inteiro, e está à distância de **12 bytes** do valor apontado por `%ebp`.
2. As estruturas de controlo possíveis são: `"if <cond> then <statement_1> else <statement_2>"; "do <statement_3> while <cond>";` ou `"while <cond> <statement_4>".`
3. Indicar claramente no *assembly* o código associado ao `<statement>`.

```

    pushl    %ebp
    movl    %esp, %ebp
    xorl    %edx, %edx
    pushl    %ebx
    movl    12(%ebp), %ecx
    xorl    %eax, %eax
    movl    8(%ebp), %ebx
.L2:
    addl    (%ebx,%edx,4), %eax
    incl    %edx
    cmpl    %ecx, %edx
    jl     .L2
    popl    %ebx
    popl    %ebp
    ret

```