ISA IA-32

Ficha 2

TeSP AM - Arquitetura de Dispositivos Móveis

André Pereira & Marco Couto

Nome:_____ Nº:_____

EXERCÍCIO I

Considere o seguinte excerto de código (em C e Assembly):

----- código C -----

```
 1 int maior(int a, int b){
 2    if(a>=b) return a;
 3    else return b;
 4 }
 5
 6 int main(){
 7    int x=1, y=-1, z=0;
 8    ... (scanf para x e y)
 9    z = maior(x,y);
11    ...
12    return 1;
13 }
```

----- código assembly -----

```
 1 maior:
 2    pushl %ebp
 3    movl %esp, %ebp
 4    movl $8(%ebp), %eax
 5    movl $12(%ebp), %ecx
 6    cmpl %ecx, %eax
 7    jl .ELSE1
 8    jmp .FIM
 9 .ELSE1
10    movl %ecx, %eax
11 .FIM
12    leave
13    ret
14
15 main:
16    ...
17    movl $1, %ebx
18    movl $-1, %ecx
19    movl $0, %eax
20    ...
21    pushl %ecx
22    pushl %ebx
23    call maior
24    addl $8, %esp
25    leave
26    ret
```

1. Porque é que na implementação em *Assembly* do ciclo tem 2 saltos seguidos (nas linhas **7** e **8**)?

2. Durante a execução da função `maior` os registos `%ecx` e `%eax` correspondem a que variáveis da função `main`?

3. A instrução `leave` modifica alguns registos. Quais são eles e como ficam depois de ela executar? E a instrução `ret`? E a instrução `call`?

4. Propõe uma implementação alternativa da função `maior`.

5. Qual o motivo de existir a instrução `addl $8, %esp`, na linha **23**?

6. Se o valor do registo `%esp` no início do programa for `0xFF0A`, qual o seu valor antes da chamada à função `maior`, antes da instrução `ret` na linha **13**, e antes da instrução `ret` na linha **26**?

EXERCÍCIO II

```
----- código C -----
 1 int soma(int a, int b){
 2     return a+b;
 3 }
 4
 5 int main(){
 6     int a=5, r=0, i;
 7     for(i=a; i>0; i--){
 8         r += soma(i,a);
 9     }
10     return 1;
11 }
```

```
----- código assembly -----
 1 soma:
 2     ____ %ebp
 3     movl %esp, %ebp
 4     movl $12(%ebp), ____
 5     ____ $8(%ebp), %eax
 6     leave
 7     ret
 8 main:
 9     ...
10     movl $0, %ebx
11     movl $5, %esi
12 LOOP1:
13     cmpl $0, %esi
14     j__ FIM1
15     pushl $5
16     pushl %esi
17     call soma
18     ____ %esi
19     addl %eax, %ebx
20     addl $8, %esp
21     jmp ____
22 FIM:
23     leave
24     ret
25
```

1. Complete os espaços em branco para que o código *Assembly* reflita exatamente o mesmo que o código **C**?

2. Se reparar, a variável a não é considerada no código *Assembly*, apenas o valor 5. Porquê?

3. Se mudar a instrução `j__ FIM1` (linha 14), como ficaria o código de maneira a que fizesse exatamente a mesma coisa?

4. É possível trocar a instrução `leave` em todos os casos em que aparece por 2 instruções que fazem exatamente a mesma coisa. Quais? O que faz cada uma?

5. Considere a instrução da linha 17 (`call soma`). Que registos são modificados após a execução desta instrução e da próxima? Que valores são introduzidos na stack? Justifique.

6. Considere que a instrução na linha 3 (`movl %esp, %ebp`) acabou de ser executada. Indique toda a informação que passa nos barramentos de endereços, dados e controlo para a execução da instrução seguinte. Se não souber o valor de algum registo use o nome desse registo.