**Arquitetura de Dispositivos Móveis**

**2015/2016 - 2º Teste**

TeSP AM - Arquitetura de Dispositivos Móveis, André Pereira

Nome:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nº:\_\_\_\_\_\_\_\_\_

Nota 1: As perguntas assinaladas com **CBi** são as de competências básicas (“**i**” indica o número da pergunta). As que estiverem assinaladas com **CCi** são perguntas de competências complementares.

Nota 2: Para todas as respostas apresente todos os cálculos efetuados

**PARTE I**

1. Considere o seguinte excerto de código C, e **complete** o código *assembly* correspondente.**(CB1)**

|  |  |
| --- | --- |
| ...  while(x < 10){  if(x<=y) x+=y;  }  ... | ...  cmpl $10, \_\_\_\_  j\_\_ .FIM  cmpl \_\_\_\_, %ebx  jl .FIM  addl \_\_\_\_, %eax  .FIM  ... |

1. **Indique** a informação que circula nos barramentos durante as 3 fases de execução (*fetch*, *decode* e *execute*) da instrução instrução movl -4(%esp), %ebx e **indique** quais os registos que foram alterados e qual o seu valor no final da instrução. Considere que a instrução em questão ocupa 16 bits em memória, e é executada numa máquina com as seguintes características: **(CB2)**
   * Arquitetura: 32 bits
   * Ordenação: *Little Endian*
   * Valores em memória:
     + De 0x7084 até 0x7087: A1 1C B9 10
     + De 0x807C até 0x8083: 1C 0A F1 2D 33 C4 01 19
   * Valores em registos:
     + %ebx = 0x0010
     + (SP) %esp = 0x8080
     + (IP) %eip = 0x7084
2. Considere o seguinte excerto de código *assembly*. **Justifique** a existência da instrução addl $12, %esp. **(CB3)**

|  |
| --- |
| ...  pushl %ebx  pushl %ecx  pushl %esi  call func  addl $12, %esp  ... |

**PARTE II**

1. **Indique**, justificando,uma instrução *assembly* que tenha um comportamento correspondente à informação que passa nos seguintes barramentos durante a sua fase de execução: **(CC1)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Execute** | | |
| **Dados** |  | 0x1010A503 | 0x1010A50A |
| **Endereços** | 0x00000100 |  | 0x00000100 |
| **Controlo** | R |  | W |

**---------------------------** tempo **---------------------------------->**

1. Considere o código em *assembly* apresentado em baixo. **Indique** o conteúdo das posições da *stack* antes da execução da instrução leave na função MENOR. Se não for possível descobrir o valor das posições deixe uma anotação (ex.: “conteúdo de %eax”). **(CC2)**

|  |
| --- |
| ...  pushl %eax  pushl %ebx  call .MENOR  addl $8, %esp  ...  .MENOR  pushl %ebp  movl %esp, %ebp  movl 8(%esp), %eax  subl 12(%esp), %eax  leave  ret |

1. Considere o código C apresentado em baixo. **Crie** o respectivo código *assembly*. **(CC3)**

|  |
| --- |
| int func (void) {  int x = 10, y = 5, z = 2;  while (x > y) y += z + 2;  return y;  } |