

**UNIVERSIDADE DE AVEIRO**  
**DEPARTAMENTO DE ELECTRONICA, TELECOMUNICAÇÕES E INFORMÁTICA**  
**Teste Teórico 2 - 4 de Julho de 2022**

Nome: \_\_\_\_\_ Nº Mec. \_\_\_\_\_

Duração: 1h15m

**Nota:** Justifique **todas** as respostas.

1) Considere os endereços de início dos segmentos: **.data** = 0x10010000 e **.text** = 0x00400000. e o trecho de código *Assembly* do MIPS apresentado.

| Linha | Label          | Assembly                              | Comentário                 |
|-------|----------------|---------------------------------------|----------------------------|
| 1     |                | <b>.data</b>                          |                            |
| 2     | <b>array:</b>  | <b>.space</b> 12                      |                            |
| 3     |                | <b>.text</b>                          |                            |
| 4     |                | <b>la</b> \$t0, <b>array</b>          | <b>#instrução virtual!</b> |
| 5     |                | <b>li</b> \$t5, 3                     |                            |
| 6     |                | <b>li</b> \$t1, 0                     |                            |
| 7     | <b>for:</b>    | <b>beq</b> \$t1, \$t5 , <b>endfor</b> |                            |
| 8     |                | <b>sll</b> \$t2, \$t1, 2              |                            |
| 9     |                | <b>addu</b> \$t2,\$t2,\$t0            |                            |
| 10    |                | <b>sw</b> \$t1,0(\$t2)                |                            |
| 11    |                | <b>j for</b>                          |                            |
| 12    | <b>endfor:</b> | <b>li</b> \$v0, 10                    |                            |
| 13    |                | <b>syscall</b>                        |                            |
| 14    |                |                                       |                            |

a) Sabendo que o opcode da instrução **beq** é 4, e que os registos \$t1 e \$t5 são os registos \$9 e \$13, apresente o código máquina (em binário) da instrução da linha 7 (**beq** \$t1, \$t5 , **endfor**).

|              |              |              |              |             |            |
|--------------|--------------|--------------|--------------|-------------|------------|
| <b>31:26</b> | <b>25:21</b> | <b>20:16</b> | <b>15:11</b> | <b>10:6</b> | <b>5:0</b> |
|              |              |              |              |             |            |

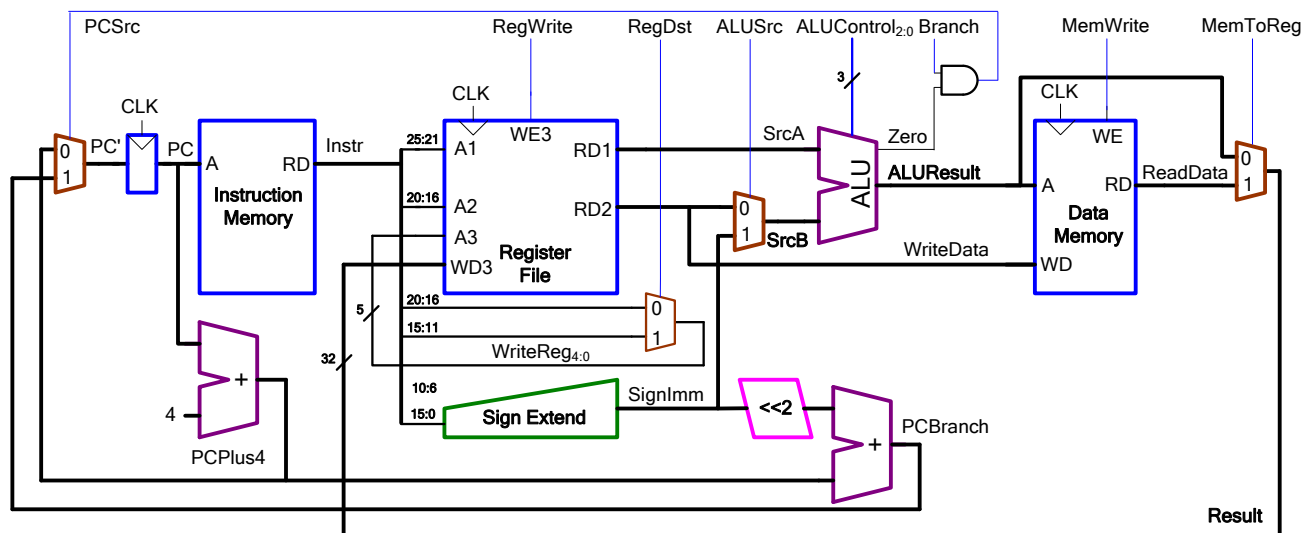
Justificação (determinação da constante):

b) Sabendo que o opcode da instrução **j** é 2, determine o código máquina (em binário) da instrução da linha 11 (**j for**).

|              |              |              |              |             |            |
|--------------|--------------|--------------|--------------|-------------|------------|
| <b>31:26</b> | <b>25:21</b> | <b>20:16</b> | <b>15:11</b> | <b>10:6</b> | <b>5:0</b> |
|              |              |              |              |             |            |

Justificação (determinação da constante):

2) A **Figura 1** representa uma implementação básica do *datapath* do MIPS.



**Figura 1 - Datapath single-cycle**

a) Preencha a tabela seguinte com o valor dos sinais de controlo durante a execução da instrução

**sw**      \$t1, 0 (\$t2)

| Sinal                      | Valores para<br><b>sw</b> \$t1, 0 (\$t2) |
|----------------------------|--|
| RegWrite                   |  |
| RegDst                     |  |
| ALUSrc                     |  |
| ALUControl (ver Tabela II) |  |
| Branch                     |  |
| MemWrite                   |  |
| MemToReg                   |  |

b) Adicione na **Figura 1** o que falta para que o *datapath* suporte a execução da instrução **sll** (*Shift Left Logical*).

Justifique/Descreva as alterações efetuadas.

c) Assinale, na Figura 1, todos os caminhos e sinais ativos durante a execução da instrução

**sll** \$t1, \$t0, 25.

d) Acrescente à Tabela de Verdade da **Tabela II** o valor das entradas e saída  $ALUControl_{2:0}$  relativas à instrução **sll**. O valor do  $FunCode_{5:0}$  de **sll** é igual a 0.

**Sugestão:** Use uma das combinações disponíveis e não altere o número de bits de  $ALUControl$ .

| $ALUOp_{1:0}$ | $Funct_{5:0}$ | $ALUControl_{2:0}$ |
|---------------|---------------|--------------------|
| 00            | XXXXXX        | 010 (Add)          |
| 01            | XXXXXX        | 110 (Subtract)     |
| 10            | 100000 (add)  | 010 (Add)          |
| 10            | 100010 (sub)  | 110 (Subtract)     |
| 10            | 100100 (and)  | 000 (And)          |
| 10            | 100101 (or)   | 001 (Or)           |
| 10            | 101010 (slt)  | 111 (SlT)          |
|               |               |                    |

Tabela II - Descodificador da ALU

e) Sendo o  $FunCode_{5:0}$  de **sll** igual a 0, preencha todos os campos de bits do quadro seguinte, para a instrução "**sll \$t1, \$t0, 25**"; o número dos registos \$t0 e \$t1 é igual a 8 e 9, respetivamente.

Instrução sll rd, rt, shamt

| 31:26 | 25:21 | 20:16 | 15:11 | 10:6 | 5:0 |
|-------|-------|-------|-------|------|-----|
|       |       |       |       |      |     |

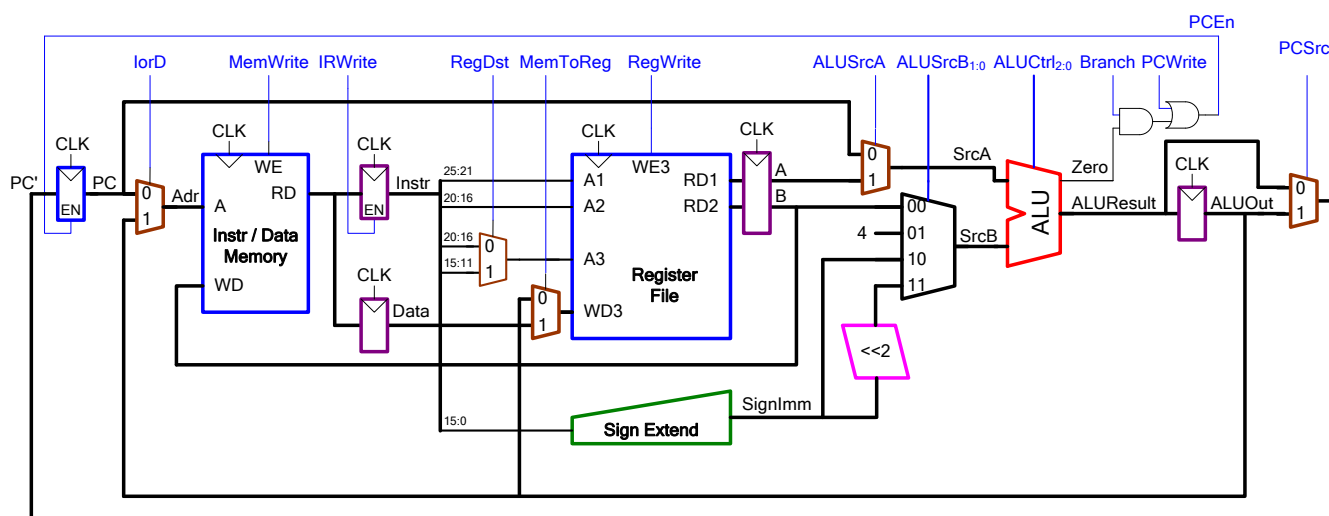
Zona de Rascunho:

f) Preencha a tabela seguinte com o valor dos sinais de controlo durante a execução da instrução:

"**sll \$t1, \$t0, 25**".

| Sinal                          | Valores para<br><b>sll \$t1, \$t0, 25</b> |
|--------------------------------|---|
| RegWrite                       |   |
| RegDst                         |   |
| ALUSrc                         |   |
| ALUControl (resposta alínea d) |   |
| Branch                         |   |
| MemWrite                       |   |
| MemToReg                       |   |

3) A **Figura 2** representa uma implementação do *datapath multicycle* e a **Figura 3** o diagrama de estados parcial do controlador respetivo.



**Figura 2 - Datapath multicycle**

a) Modifique, se necessário, o *datapath* da **Figura 2** para suportar a execução da instrução **andi**.

Justifique a(s) alteração(ões) efetuada(s).

b) Indique, na **Tabela III**, as alterações necessárias no decodificador da ALU.

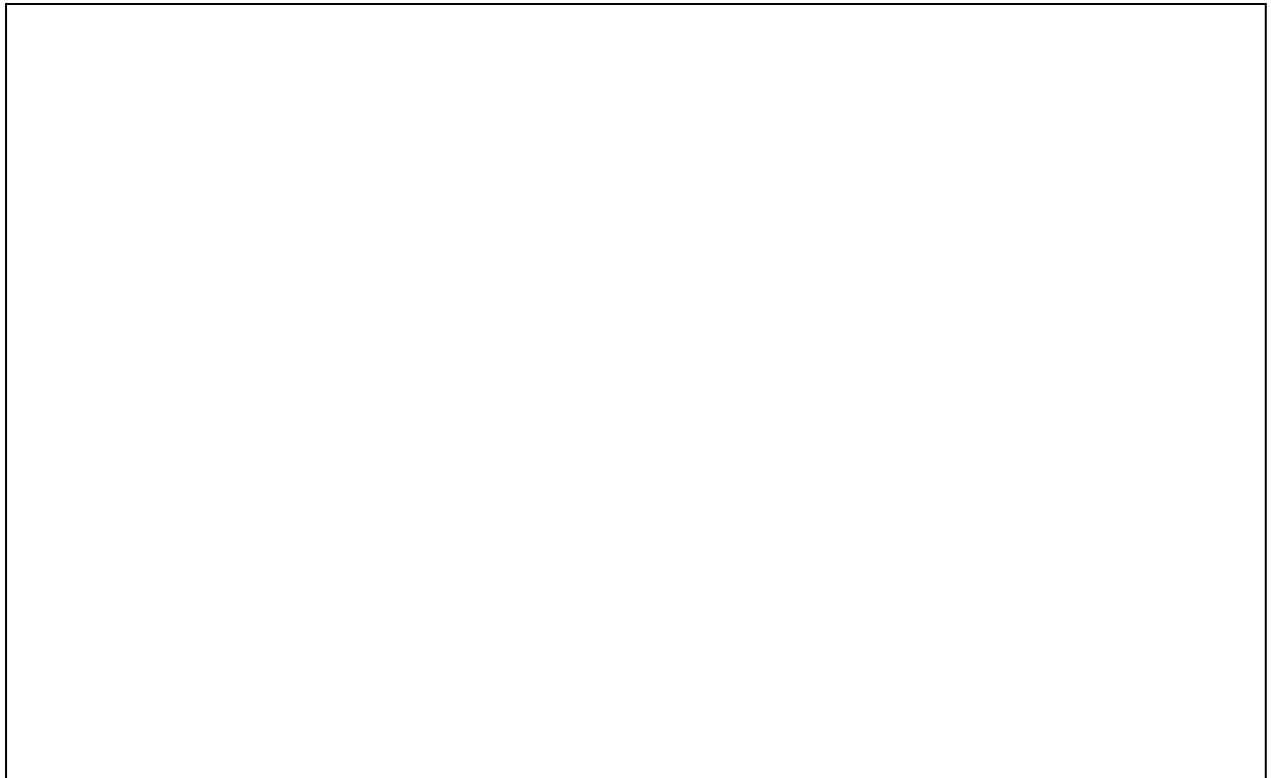
**Sugestão:** Use uma das combinações disponíveis e não altere o número de bits de *ALUControl*.

| ALUOp <sub>1:0</sub> | Funct <sub>5:0</sub> | ALUControl <sub>2:0</sub> |
|----------------------|----------------------|---------------------------|
| 00                   | XXXXXX               | 010 (Add)                 |
| 01                   | XXXXXX               | 110 (Subtract)            |
| 10                   | 100000 (add)         | 010 (Add)                 |
| 10                   | 100010 (sub)         | 110 (Subtract)            |
| 10                   | 100101 (or)          | 001 (Or)                  |
| 10                   | 100110 (xor)         | 100 (Xor)                 |
| 10                   | 101010 (slt)         | 111 (SlT)                 |
|                      |                      |                           |

**Tabela III - Decodificador da ALU, multi cycle.**

- c) Identifique cada um dos estados percorridos pela unidade de controlo durante a execução da instrução **andi** e explique por palavras as ações envolvidas em cada um deles.

Sugestão: Consulte o diagrama de estados da Figura 3.



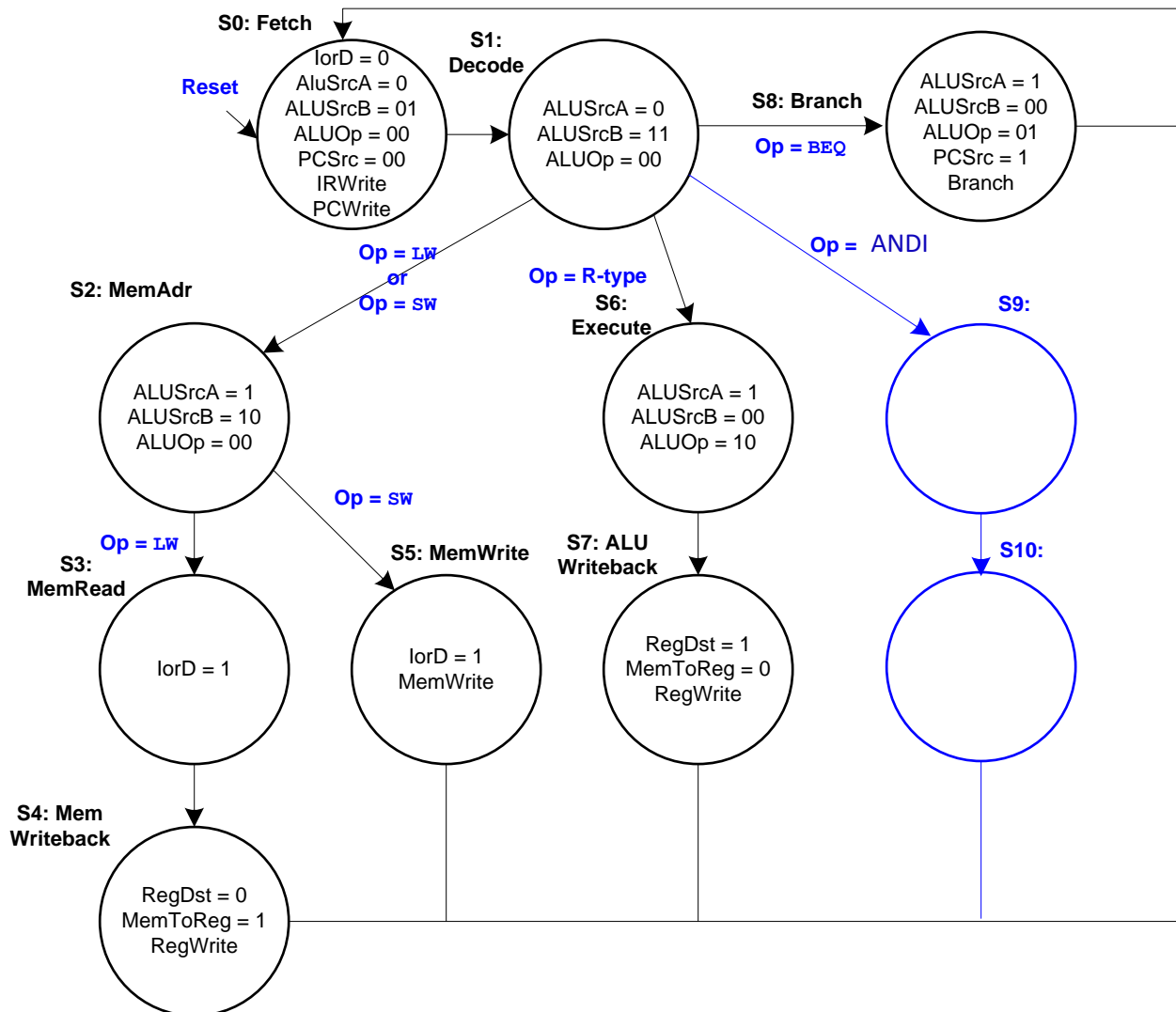
- d) Considere nesta alínea, os caminhos ativos e o valor dos sinais de controlo relevantes durante a fase EXECUTE, da instrução **andi**.

Preencha na tabela seguinte o valor que os sinais de controlo assumem durante esse estado e assinale na figura 2 os caminhos ativos nessa fase.

| Instr.      | Op <sub>5:0</sub> | lor | MemWrite | RegDst | MemToReg | RegWrite | AluSrcA | AluSrcB | Branch | PCSrc | ALUCtrl |
|-------------|-------------------|-----|----------|--------|----------|----------|---------|---------|--------|-------|---------|
| <b>andi</b> | 001100            |     |          |        |          |          |         |         |        |       |         |

Sinais de Controlo na fase EXECUTE da instrução andi

e) No diagrama de estados da **Figura 3** preencha os estados **S9** e **S10** com os valores dos sinais de seleção dos *multiplexers* e de *enable*. Use o decodificador da ALU da **Tabela III**.



**Figura 3 - Diagrama de Estados do Controlador Principal**

**4) Traduza para assembly a seguinte função print\_fact.**

Respeite a convenção de uso de registos abordada nas aulas.

Não precisa de implementar a função **factorial**, basta invocá-la de acordo com a convenção.

```
int factorial(int );
int print_fact(int n){
    int i;
    if (n < 0)
        return -1;
    else
    {
        for(i = 0; i <= n; i++)
            print_int10( factorial(i) );
    }
    return 1;
}
```

| Variável | Registo(s) |
|----------|------------|
| n        |            |
| i        |            |
|          |            |
|          |            |

[illegible]