

Nome: _____

N. Mec: _____

Grupo I

Responda às questões seguintes, **mostrando todos os passos da sua resposta**:

a) O valor dos 16 bits menos significativos, expresso em hexadecimal, do código máquina da instrução **L1: beq \$1,\$1,L1** é:

b) Suponha que **\$2=0xA35** e que se pretende aceder, através da instrução **LW**, ao endereço de memória **0xA31**. Para que isso aconteça, o valor dos 16 bits menos significativos, expresso em hexadecimal, do código máquina da instrução **"lw \$3,??(\$2)"** deve ser:

c) O valor **0xAC640100** é o código máquina de uma instrução do MIPS. Apresente a instrução *Assembly* completa a que corresponde esse código (mnemónica e argumentos – consulte a tabela de códigos disponível no Grupo III):

d) Admita que os valores indicados no *datapath* da Figura 1 correspondem à “fotografia” tirada no decurso da execução de uma dada instrução. Observe todos os sinais e valores presentes nessa figura e responda às seguintes questões:

1) A fase de execução em que se encontra é _____, porque _____

2) A instrução que se encontra em execução é (não necessita de colocar a instrução completa) _____

_____, porque _____

3) A instrução que vai ser executada de seguida encontra-se no endereço _____, porque _____

4) Admita que o sinal **"RegWrite"** tinha o valor '0'. Podia então concluir-se que a instrução em execução estava na fase _____, porque _____

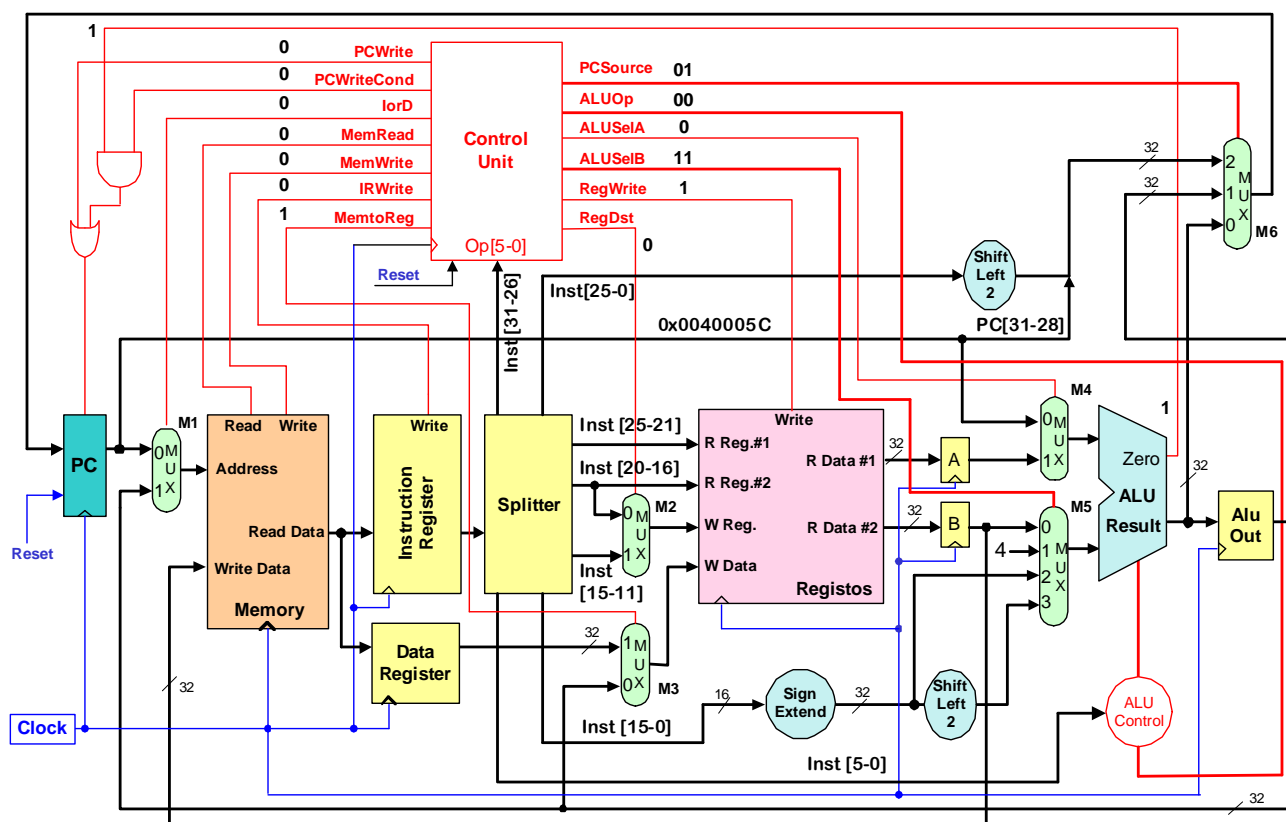


Figura 1. Datapath multi-cycle do MIPS.

Zona de rascunho

Nome: _____

N. Mec: _____

Grupo II

Considere o *datapath* e a unidade de controlo fornecidos na Figura 1, sabendo que corresponde a uma implementação *multi-cycle* simplificada do MIPS, sem *pipelining*.

- a) Preencha a tabela seguinte com o nome de cada uma das fases de execução da instrução "**xor \$1,\$2,\$3**" e com o valor que tomam, em cada uma delas, os sinais de controlo ali indicados. Admita que o valor lógico "1" corresponde ao estado ativo. **Assinale as situações de "don't care" com "X".**

Designação da Fase	Instruction Fetch/Calc. PC+4				
PCWrite					
MemWrite					
IRWrite					
ALUOp					
ALUSelA					
ALUSelB					
lorD					
PCSource					
MemRead					
RegWrite					
RegDst					

- b) Preencha a tabela seguinte com o nome de cada uma das fases de execução da instrução "**xor \$1,\$2,\$3**" (código máquina **0x00430826**) e com o valor que tomam, em cada uma delas, os valores do *datapath* ali indicados. Considere que os registos, no instante em que vai iniciar-se o *instruction fetch*, têm os seguintes valores: **\$1=0x145**, **\$2=0x3A4**, **\$3=0x75D**, **PC=0x0040008C**). **Assinale as situações de "valor desconhecido" com "?".**

Designação da fase	Instruction Fetch/Calc. PC+4				
PC	0x0040008C				
Instr. Register					
Data Register					
A					
B					
ALU Result					
ALU Out					
ALU Zero					

ALUOp – 00: Add, 01: Subtract, 10: R-Type, 11: Set if Less Than

Nome:

N. Mec:

Grupo III

Apresente todos os passos que justifiquem as suas respostas: a simples apresentação de valores sem justificação adequada terá cotação 0

Considere o trecho de código da tabela ao lado onde o endereço representado pelo *label* L1 é 0x0000A204.

- a) Traduza para código máquina do MIPS as instruções abaixo indicadas (expressando o resultado em hexadecimal) e indique o endereço de memória em que se encontra cada uma. Mostre todos os passos da sua resposta e, no final, preencha a tabela.

sw \$4, 0x200(\$3)

Opcode	Funct	Instr.
0	0x20	add
0	0x26	xor
0x04		beq
0x2B		sw
0x02		j
0x08		addi
0x0A		slti
0x23		lw

L1:	addi \$2,\$0,0x14
	add \$3,\$0,\$0
L2:	beq \$3,\$2,L3
	lw \$4,0x100(\$3)
	sw \$4,0x200(\$3)
	addi \$3,\$3,4
	j L2
L3:	...

j L2

Endereço	Instrução	Código Máquina (hexadecimal)
	sw \$4, 0x200(\$3)	
	j L2	

- b) Calcule o número total de ciclos de relógio que demora a execução completa desse trecho de código (desde o instante inicial do *instruction fetch* da primeira instrução até ao momento em que vai iniciar-se o *instruction fetch* da instrução presente em "L3:"): i) num datapath single-cycle; ii) num datapath multi-cycle. **Apresente todos os passos que justifiquem a sua resposta** (a simples apresentação de valores sem justificação adequada terá cotação 0).

i) *Single-cycle*:

ii) *Multi-cycle*:

Nome: _____

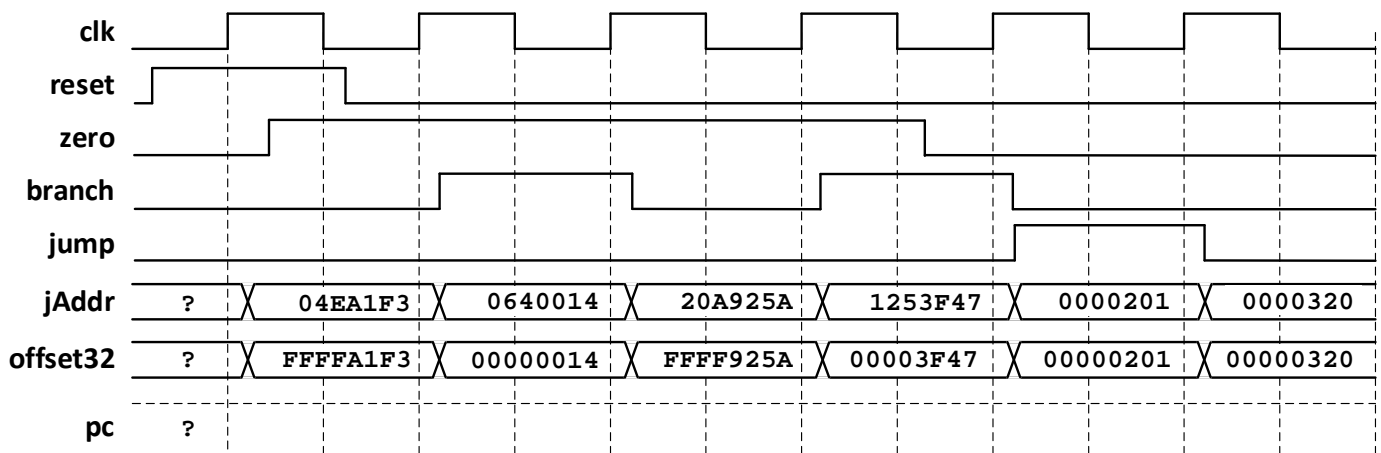
N. Mec: _____

Grupo IV

- a) O código VHDL que se apresenta de seguida corresponde a uma possível implementação do módulo "PC_update" para a arquitetura *single-cycle* do MIPS que implementou nas aulas práticas, responsável pela manutenção e atualização do valor do *Program Counter*. Complete o diagrama temporal da figura seguinte, calculando o valor de saída ("**pc**") para todos os ciclos de relógio ali apresentados (note que os valores de "**jAddr**" e "**offset32**" estão representados em hexadecimal).

```
entity PC_update is
    port(clk, reset, branch, jump, zero : in std_logic;
         offset32 : in std_logic_vector(31 downto 0);
         jAddr    : in std_logic_vector(25 downto 0);
         pc       : out std_logic_vector(31 downto 0));
end PC_update;

architecture Behavioral of PC_update is
    signal s_pc, s_offsetSL2 : unsigned(31 downto 0);
    signal s_pc4 : unsigned(31 downto 0);
begin
    s_offsetSL2 <= unsigned(offset32(29 downto 0)) & "00";
    s_pc4 <= s_pc + 4;
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(reset = '1') then
                s_pc <= (others => '0');
            else
                if(jump = '1') then
                    s_pc <= s_pc4(31 downto 28) & unsigned(jAddr) & "00";
                elsif(branch = '1' and zero = '1') then
                    s_pc <= s_pc4 + s_offsetSL2;
                else
                    s_pc <= s_pc4;
                end if;
            end if;
        end if;
    end process;
    pc <= std_logic_vector(s_pc);
end Behavioral;
```



a) Complete o código VHDL seguinte com a implementação de um registo de N bits com reset síncrono e *enable*. A ação de reset não deve depender do sinal *enable*.

architecture _____ of _____ is

[illegible]

```
end behav;
```

Zona de rascunho