Desafio do Trabalho de Organização de Computadores
- Criação de Instrução: XOR -

Base:

- XOR (Exclusivo) é uma instrução do Tipo-R (como ADD/SUB);
- No ula_op já usamos 00, 01, 10 e temos o **11** disponível pra ela;
- Opcode**:** Vamos usar **`0011`** (já que 0000=ADD, 0001=AND, 0010=SUB);

Passo a passo:

1. Modificar a ULA:

```
architecture Behavioral of ula is
begin
    process(a, b, ula_op)
        variable res : signed(31 downto 0);
    begin
        case ula_op is
            when "00" =>   -- ADD
                res := signed(a) + signed(b);

            when "01" =>   -- SUB
                res := signed(a) - signed(b);

            when "10" =>   -- AND
                res := signed(a and b);

            -- NOVO: Instrução XOR (Opcode 11)
            when "11" =>
                res := signed(a xor b);

            when others =>
                res := (others => '0');
        end case;

        result <= std_logic_vector(res);
    end process;
end Behavioral;
```

codesnap.dev

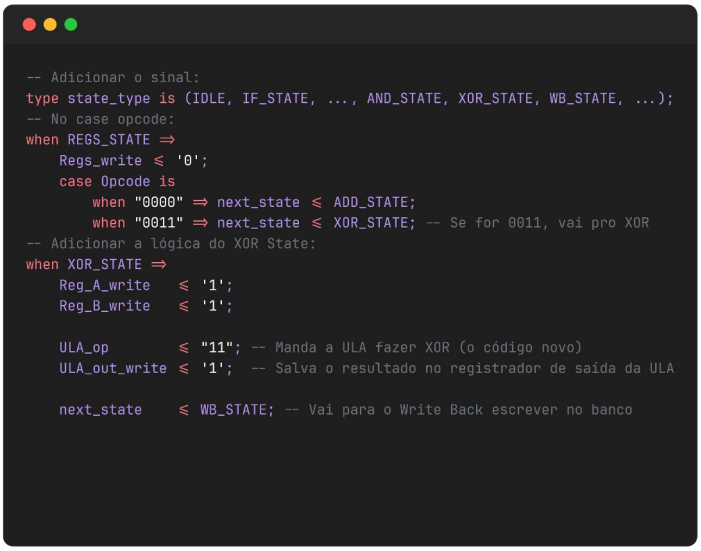
2. Modificar o Datapath:
   - Toda vez que o processador ver o opcode `0011`, ele tem que saber que deve escrever no registrador de destino **RD** (bits 15-11).

```
process(opcode, rs, rt, rd)
begin
    case opcode is
        when "0000" | "0001" | "0010" | "0011" => -- Adicionado o 0011 (XOR)
            write_reg <= rd;

        when "0100" => -- LW
            write_reg <= rt;

        when others =>
            write_reg <= (others => '0');
    end case;
end process;
```


3. Modificar a Unidade de Controle:



```
-- Adicionar o sinal:
type state_type is (IDLE, IF_STATE, ..., AND_STATE, XOR_STATE, WB_STATE, ...);
-- No case opcode:
when REGS_STATE =>
    Regs_write <= '0';
    case Opcode is
        when "0000" => next_state <= ADD_STATE;
        when "0011" => next_state <= XOR_STATE; -- Se for 0011, vai pro XOR
-- Adicionar a lógica do XOR State:
when XOR_STATE =>
    Reg_A_write   <= '1';
    Reg_B_write   <= '1';

    ULA_op        <= "11"; -- Manda a ULA fazer XOR (o código novo)
    ULA_out_write <= '1';  -- Salva o resultado no registrador de saída da ULA

    next_state    <= WB_STATE; -- Vai para o Write Back escrever no banco
```


4. Corrigir os possíveis bugs!