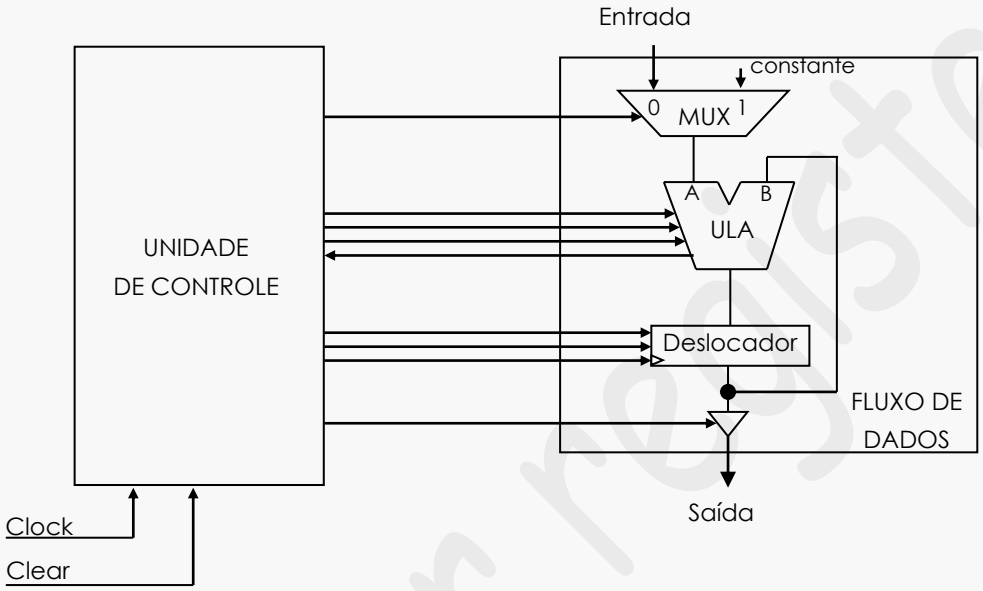
FLUXO DE DADOS "DATAPATH"

1. Introdução: É um conjunto de unidades funcionais destinadas à solução de problemas. O fluxo de dados contém unidades de multiplexagem, de armazenamento, de operações lógicas e aritméticas e opera sob gerência e supervisão de uma unidade de controle a qual gera sinais de controle através das microoperações sequenciais.

1.1 **DATAPATH** – A estrutura a seguir mostra o fluxo de dados e a unidade de controle.



2. PROCESSO DE PROJETO CIRCUITOS DIGITAIS

O processo de projeto de circuitos digitais pode ser descrito em duas fases sendo captura e conversão para circuito. Por exemplo:

Tipo de circuito	Captura	Conversão
Combinatório	Comportamento do circuito por: Equação booleana ou tabela da verdade	Converte o comportamento Para circuito.
Seqüencial	Comportamento do circuito por: equação de estados e saída (FSM)	Converte o comportamento Para circuito.
RTL	Comportamento do processador por: FSM de alto nível RTL	Converte o comportamento Para circuito.

3. MÉTODO DE PROJETO DO RTL.

O método de projeto RTL será realizado, conforme acima em quatro passos como descritos a seguir na tabela a seguir.

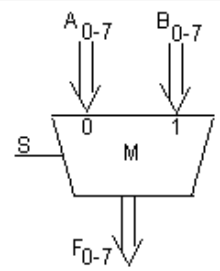
Passo 1: Processo de captura o qual descreve o comportamento do circuito através de um algoritmo com operações a serem realizadas dentro de uma seqüência desejada;
Passo 2: Definição do fluxo de dados;
Passo 3: Projeto do controlador ou unidade de controle;

Passo 4: Conversão do algoritmo em instruções da FSM de alto nível e definindo as palavras de controle a serem geradas pela unidade de controle ao fluxo de dados e os sinais recebidos pela unidade de controle do fluxo de dados.

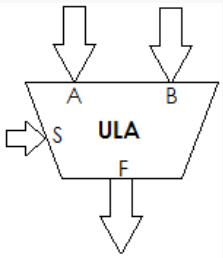
Obs.: Um quinto passo é necessário para definir a freqüência do relógio e o tipo de transição das unidades funcionais, subida ou descida.

5. Unidades funcionais – As unidades funcionais do fluxo de dados são:

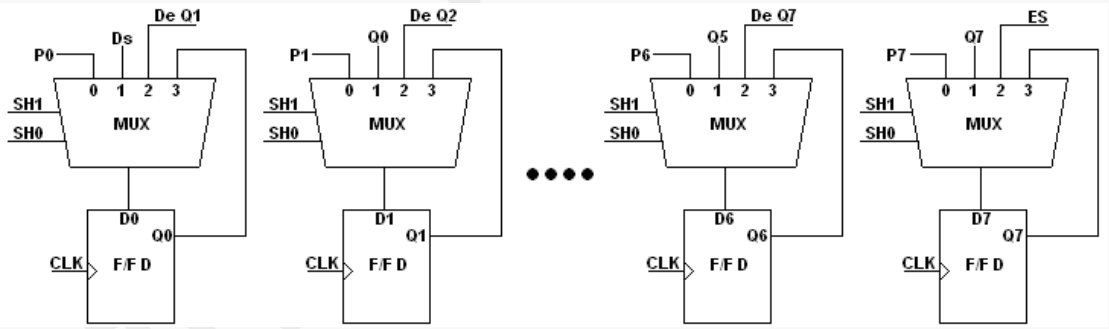
5.a Multiplexador – Unidade 2 x 1 controlada pela variável de controle S.



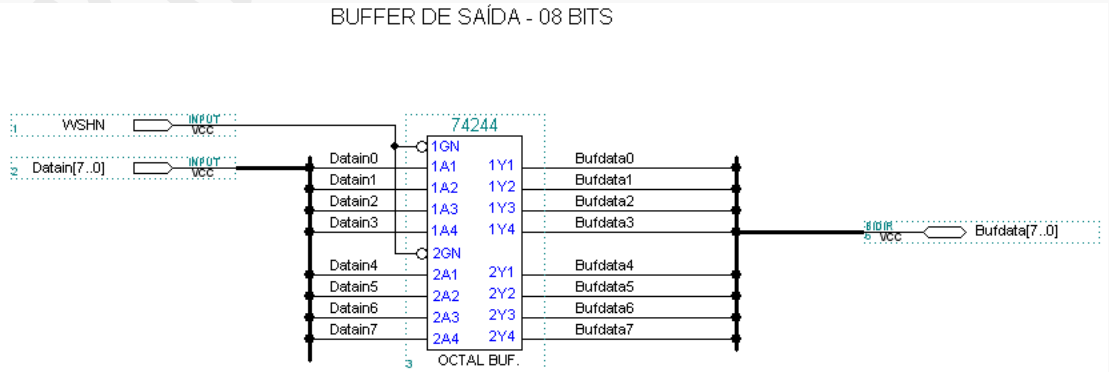
5.b – Unidade lógica e aritmética.



5.c – Deslocador.



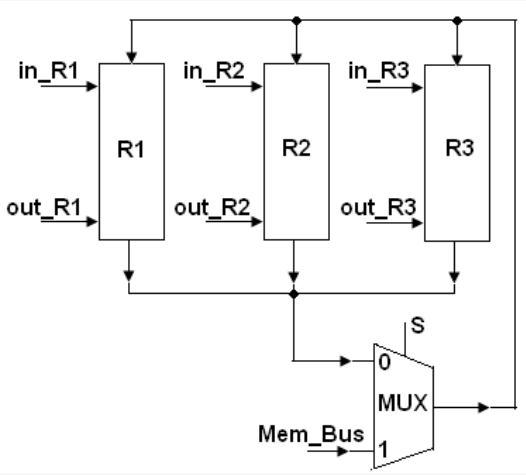
5.d – Buffer de saída.



6. Unidade de controle – Responsável pela geração e controle sequencial síncrono das operações realizadas no fluxo de dados. É projetada como uma máquina de estados finitos a partir de um problema. Gera os sinais de controle e recebe "status" do fluxo de dados.

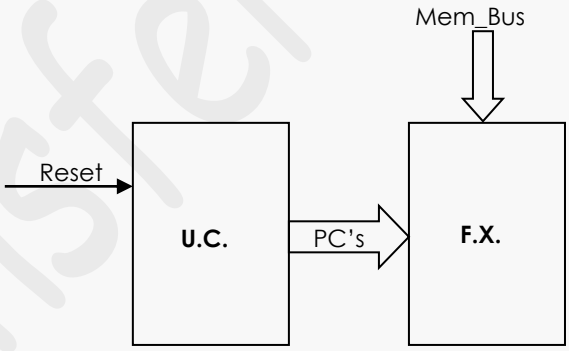
Exemplo: Transferência entre registradores de dados executar na seqüência as instruções:

1. Mem_Bus ⟶ R_1
2. R_3 ⟶ R_2
3. Mem_Bus ⟶ R_3

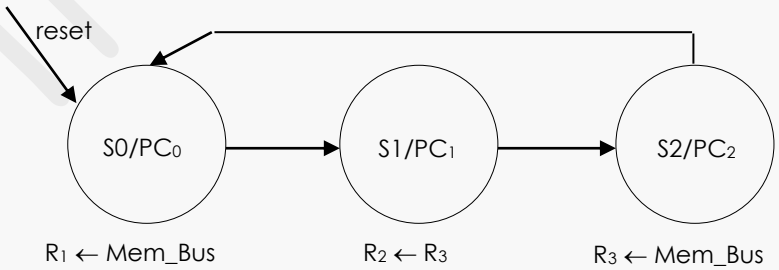


Pede-se:
a)Representação esquemática do sistema digital completo.
b) Projeto da UC
c) Quadro de instruções.

a) Representação esquemática do sistema digital completo.



b) Diagrama de estado de descrição da F.S.M. da U.C.



c) Quadro de instruções

item	instrução	S	in_R_1	in_R_2	in_R_3	out_R_1	out_R_2	out_R_3	PC's
1	$R_1 \leftarrow$ Mem_Bus	1	1	0	0	0	0	0	PC_0
2	$R_2 \leftarrow R_3$	0	0	1	0	0	0	1	PC_1
3	$R_3 \leftarrow$ Mem_Bus	1	0	0	1	0	0	0	PC_2

d) Implementação da F.S.M.

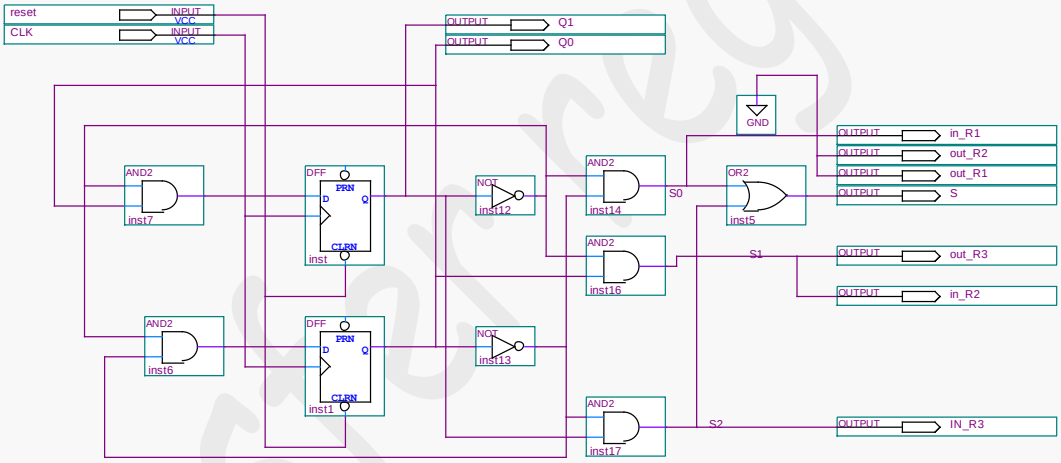
Atual	Futuro	-
Q_1Q_0	Q_1Q_0	PC's
S0 - 00	01	PC_0
S1 - 01	10	PC_1
S2 - 10	00	PC_2
S3 - 11	-	-

Equações booleanas entradas/saídas.

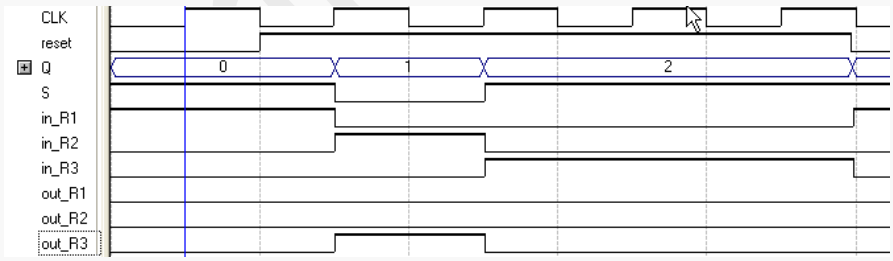
$D_1 = Q_1'Q_0$
$D_0 = Q_0'Q_1'$
$PC_0 = (60)_H$
$PC_1 = (11)_H$
$PC_2 = (48)_H$

$S = S0 + S2;$
$in\_R_1 = S0;$
$in\_R_2 = S1;$
$in\_R_3 = S_2;$
$out\_R_1 = 0;$
$out\_R_2 = 0;$
$out\_R_3 = S1$

e) Circuito final da U.C.



f) Formas de ondas da U.C.



7. Parte final:

VHDL – FLUXO DE DADOS

```
-- Quartus II VHDL
-- Basico latch
-- Registradores temporarios
library ieee;
use ieee.std_logic_1164.all;

entity  reg_3                   is port                                 (
                        in_R1               : in bit;
                        in_R2               : in bit;
                        in_R3               : in bit;
                        D                               : in std_logic_vector  (7 downto 0);
                        Q1                              : out std_logic_vector (7 downto 0);
                        Q2                              : out std_logic_vector (7 downto 0);
                        Q3                              : out std_logic_vector (7 downto 0));

end reg_3;

architecture Behavioral of reg_3 is

        signal dado1,dado2,dado3: std_logic_vector (7 downto 0);

begin

        process (in_R1,in_R2,in_R3)
                Variable valor: Integer range 0 to 255;
                begin

                        IF in_R1 = '1' then dado1 <= D;
                        ELSE dado1 <= dado1;
                        END IF;
                        IF in_R2 = '1' then dado2 <= D;
                        ELSE dado2 <= dado2;
                        END IF;
                        IF in_R3 = '1' then dado3 <= D;
                        ELSE dado3 <= dado3;
                        END IF;
        End Process;
        Q1 <= dado1;
        Q2 <= dado2;
        Q3 <= dado3;
End Behavioral;

-- 2-to-1 MUX
LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY mux2 IS PORT (
        S0                      : IN std_logic; -- select line
        D0,D1       : IN std_logic_vector(7 downto 0); -- data bus input
        Y: OUT std_logic_vector(7 downto 0)); -- data bus output
        END mux2;
```

```
ARCHITECTURE Behavioral OF mux2 IS
        BEGIN
                PROCESS(S0,D1,D0)
        BEGIN
        IF (S0='0')THEN
        Y <= D0;
        END IF;
        IF (S0 ='1') THEN
        Y <= D1;
        END IF;

                END PROCESS;
END Behavioral;

 -- Tri-state buffer
LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY buffer1 IS PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 DOWNTO 0);
        saida: OUT std_logic_vector(7 DOWNTO 0));
        END buffer1;

ARCHITECTURE Behavioral OF buffer1 IS
        BEGIN
                PROCESS (OE, D) -- get error message if no d
                        BEGIN
                        IF (OE = '1') THEN
                        saida <= D;
                        ELSE
                        saida <= (OTHERS => 'Z'); -- to get 8 Z values
                        END IF;
                END PROCESS;
        END Behavioral;
 -- Tri-state buffer
LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY buffer2 IS PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 DOWNTO 0);
        saida: OUT std_logic_vector(7 DOWNTO 0));
        END buffer2;

ARCHITECTURE Behavioral OF buffer2 IS
        BEGIN
                PROCESS (OE, D) -- get error message if no d
                        BEGIN
                        IF (OE = '1') THEN
                        saida <= D;
                        ELSE
                        saida <= (OTHERS => 'Z'); -- to get 8 Z values
                        END IF;
                END PROCESS;
        END Behavioral;
```

```
 -- Tri-state buffer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY buffer3 IS PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 DOWNTO 0);
        saida: OUT std_logic_vector(7 DOWNTO 0));
        END buffer3;

ARCHITECTURE Behavioral OF buffer3 IS
                BEGIN
                        PROCESS (OE, D) -- get error message if no d
                                BEGIN
                                IF (OE = '1') THEN
                                saida <= D;
                                ELSE
                                saida <= (OTHERS => 'Z'); -- to get 8 Z values
                                END IF;
                        END PROCESS;
        END Behavioral;

 -- BUS CANAL
LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY Bus_canal IS PORT (
        out_R1: in std_logic;
        out_R2: in std_logic;
        out_R3: in std_logic;
        A: in std_logic_vector(7 downto 0);
        B: in std_logic_vector(7 downto 0);
        C: in std_logic_vector(7 downto 0);
        Y: out std_logic_vector(7 DOWNTO 0));
        END Bus_canal;

ARCHITECTURE Behavioral OF Bus_canal IS
                BEGIN
                        PROCESS(out_R1,out_R2,out_R3)
                        BEGIN
                        IF out_R1 ='1' Then Y <= A;
                        ELSIF out_R2 = '1' Then Y <=B;
                        ELSIF out_R3 = '1' Then Y <=C;
                        ELSE
                        Y <= (OTHERS =>'Z');
                        END IF;
                        END PROCESS;
                END Behavioral;
-- Tri-state buffer
LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY apresenta IS PORT (
        OE: IN std_logic;
D: IN std_logic_vector(7 DOWNTO 0);
        output: OUT std_logic_vector(7 DOWNTO 0));
        END apresenta;
```

```
ARCHITECTURE Behavioral OF apresenta IS
                BEGIN
                        PROCESS (OE, D) -- get error message if no d
                                BEGIN
                                IF (OE = '1') THEN
                                output <= D;
                                ELSE
                                output <= (OTHERS => 'Z'); -- to get 8 Z values
                                END IF;
                        END PROCESS;
        END Behavioral;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
        ENTITY rtl_aula15 IS PORT (
        input_mem: IN std_logic_vector(7 DOWNTO 0 );
        IE0: IN std_logic;
        in_R1: in bit;
        in_R2: in bit;
        in_R3: in bit;
        out_R1:in std_logic;
        out_R2:in std_logic;
        out_R3:in std_logic;
        OE: in std_logic;
        output: out std_logic_vector(7 downto 0));
        END rtl_aula15;

        ARCHITECTURE Structural OF rtl_aula15 IS

        COMPONENT reg_3 PORT(
        in_R1,in_R2,in_R3: in bit;
        D: in std_logic_vector(7 DOWNTO 0);
        Q1: out std_logic_vector(7 DOWNTO 0);
        Q2: out std_logic_vector(7 DOWNTO 0);
        Q3: out std_logic_vector(7 DOWNTO 0));
        END COMPONENT;

        COMPONENT mux2 PORT (
        S0: IN std_logic; -- select lines
        D0, D1: IN std_logic_vector(7 DOWNTO 0); -- data bus input
        Y: OUT std_logic_vector(7 DOWNTO 0)); -- data bus output
        END COMPONENT;

        COMPONENT buffer1 PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 downto 0);
        Saida: OUT std_logic_vector(7 downto 0));
        END COMPONENT;

        COMPONENT buffer2 PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 downto 0);
        Saida: OUT std_logic_vector(7 downto 0));
        END COMPONENT;
```

```
        COMPONENT buffer3 PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 downto 0);
        Saida: OUT std_logic_vector(7 downto 0));
        END COMPONENT;

        COMPONENT Bus_canal PORT(
        out_R1: in std_logic;
        out_R2: in std_logic;
        out_R3: in std_logic;
        A: in std_logic_vector(7 downto 0);
        B: in std_logic_vector(7 downto 0);
        C: in std_logic_vector(7 downto 0);
        Y: out std_logic_vector(7 downto 0));
        END COMPONENT;

        COMPONENT apresenta PORT (
        OE: IN std_logic;
        D: IN std_logic_vector(7 downto 0);
        output: OUT std_logic_vector(7 downto 0));
        END COMPONENT;

        SIGNAL muxout : std_logic_vector(7 DOWNTO 0 );
        SIGNAL tristateout1, tristateout2,tristateout3: std_logic_vector(7 DOWNTO 0 );
        SIGNAL R1_out, R2_out,R3_out: std_logic_vector(7 downto 0);
        SIGNAL orout: std_logic_vector(7 downto 0);
        BEGIN
-- doing structural modeling here
        U0: reg_3 PORT MAP(in_R1,in_R2,in_R3,muxout,R1_out,R2_out,R3_out);
        U1: mux2 PORT MAP(IE0,input_mem,orout,muxout );
        U2: buffer1 PORT MAP(out_R1, R1_out,tristateout1);
        U3: buffer2 PORT MAP(out_R2, R2_out,tristateout2);
        U4: buffer3 PORT MAP(out_R3, R3_out,tristateout3);
        U5: Bus_canal PORT MAP(out_R1,out_R2,out_R3,tristateout1,tristateout2,tristateout3,orout);
        U6: apresenta PORT MAP(OE,muxout,output);
        END Structural;

VHDL – Unidade de controle

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY uc_aula15 IS
        PORT(
                clk                             : IN BIT;
                reset                   : IN BIT;
                q                               : OUT BIT_VECTOR(1 DOWNTO 0);
                B                               : OUT std_logic_vector(6 downto 0));
END uc_aula15;

ARCHITECTURE exemplo OF uc_aula15 IS
        TYPE maq_estado IS (carga,transfer,move);
        SIGNAL state: maq_estado;
BEGIN
        PROCESS (clk)
```

```
BEGIN
                IF reset = '1'
                                        Then state <= carga;

        ELSIF clk'EVENT AND clk = '1' THEN
                        CASE state IS
                                WHEN carga =>
                                        state <= transfer;
                                WHEN transfer =>
                                        state <= move;
                                WHEN move =>
                                state <= carga;
                        END CASE;
                END IF;
                        CASE state IS
                                WHEN carga      => B <= "1100000";
                                WHEN transfer   => B <= "0010001";
                                WHEN move               => B <= "1001000";
                        END CASE;
        END PROCESS;
                        WITH state SELECT
                        q <="00" WHEN   carga,
                                "01" WHEN       transfer,
                                "10" WHEN       move;
                         END exemplo;
```

VHDL – Associação UC + DATAPATH

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY cpu_aula15 IS PORT (
clk_cpu: in std_logic;
reset_cpu: IN std_logic;
input_cpu: in std_logic_vector(7 downto 0);
output_cpu: out std_logic_vector(7 downto 0));
END cpu_aula15;

ARCHITECTURE structure OF cpu_aula15 IS

COMPONENT uc_aula15 PORT (
clk             : in std_logic;
reset   : in std_logic;

IE0                             : out std_logic;
in_R1                   : out std_logic;
in_R2                   : out std_logic;
in_R3                   : out std_logic;
out_R1                  : out std_logic;
out_R2                  : out std_logic;
out_R3                  : out std_logic;
OE                              : out std_logic);

END COMPONENT;
```

```
COMPONENT rtl_aula15 PORT (

input_mem: in std_logic_vector(7 DOWNTO 0 );
IE0          : in std_logic;
in_R1 : in std_logic;
in_R2 : in std_logic;
in_R3 : in std_logic;
out_R1: in std_logic;
out_R2: in std_logic;
out_R3: in std_logic;
OE           : in std_logic;
output: out std_logic_vector(7 downto 0));
END COMPONENT;

--SIGNAL C_input_mem : std_logic_vector(7 DOWNTO 0);
SIGNAL output: std_logic_vector(7 downto 0);
SIGNAL C_IE0,C_out_R1,C_out_R2,C_out_R3,C_OE: std_logic;
SIGNAL C_in_R1,C_in_R2,C_in_R3: std_logic;

BEGIN
U0: uc_aula15 PORT MAP(clk_cpu,reset_cpu,C_IE0,C_in_R1,C_in_R2,C_in_R3,C_out_R1,C_out_R2,C_out_R3,C_OE);
U1: rtl_aula15 PORT MAP(input_cpu,C_IE0,C_in_R1,C_in_R2,C_in_R3,C_out_R1,C_out_R2,C_out_R3,C_OE,output_cpu);

END structure;
```