```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity data_path is
   port(
      --clock and memory clock
      clk, mclk : in std_logic;
      --memory signals
      wen, en: in std_logic;
      --register control signals
      clr_a, ld_a: in std_logic;
      clr_b, ld_b: in std_logic;
      clr_c, ld_c: in std_logic;
      clr_z, ld_z: in std_logic;
      clr_pc, ld_pc: in std_logic;
      clr_ir, ld_ir: in std_logic;
      --register outputs
      out_a, out_b: out std_logic_vector(31 downto 0);
      out_c, out_z: out std_logic;
      outp, out_ir: out std_logic_vector(31 downto 0);
      --pc special input
      inc_pc: in std_logic;
      --address and data bus signal debug
      addr_out : inout std_logic_vector(31 downto 0);
      data_in: in std_logic_vector(31 downto 0);
      DATA_BUS, mem_out, mem_in: inout std_logic_vector(31 downto 0);
      in_mux1: in std_logic;
      in_mux2: in std_logic_vector(1 downto 0);
      --alu op
      alu_op: in std_logic_vector(2 downto 0);
      --mux controllers
      data_mux : in std_logic_vector(1 downto 0);
      reg_mux: in std_logic;
      a_mux, b_mux: in std_logic;
      im_mux1, im_mux2: in std_logic_vector(1 downto 0));
end data_path;

architecture description of data_path is
   --components

   component alu
      port(
         a, b: in std_logic_vector(31 downto 0);
         op: in std_logic_vector(2 downto 0);
         result: inout std_logic_vector(31 downto 0);
         cout: out std_logic;
         zero: out std_logic);
   end component alu;

   component register1
      port(
         d,ld,clr,clk: in std_logic;
         Q: out std_logic);
   end component register1;

   component register32
      port(
         d: in std_logic_vector(31 downto 0);
         ld,clr,clk: in std_logic;
         Q: out std_logic_vector(31 downto 0));
   end component register32;

   component pc
      port(
         clr, clk, ld, inc: in std_logic;
         d: in std_logic_vector(31 downto 0);
         q: inout std_logic_vector(31 downto 0));
   end component pc;

   component memoryModule
      port (
         clk: in std_logic;
         addr: in unsigned (7 downto 0);
         data_in : in std_logic_vector(31 downto 0);
         wen, en: in std_logic;
         data_out: out std_logic_vector(31 downto 0));
   end component memoryModule;

   component lze
      port(
         lze_in: in std_logic_vector(31 downto 0);
         lze_out: out std_logic_vector(31 downto 0));
   end component lze;

   component uze
      port(
         uze_in : in std_logic_vector(31 downto 0);
         uze_out: out std_logic_vector(31 downto 0));
   end component uze;

   component red
      port(
         red_in : in std_logic_vector(31 downto 0);
         red_out : out unsigned(7 downto 0));
   end component red;

   component mux2to1
      port (
         s : in std_logic;
         w0, w1 : in std_logic_vector(31 downto 0);
         f: out std_logic_vector(31 downto 0));
   end component mux2to1;

   component mux4to1
      port (
         s : in std_logic_vector(1 downto 0);
         x0, x1, x2, x3 : in std_logic_vector(31 downto 0);
         f : out std_logic_vector(31 downto 0));
   end component mux4to1;

   --signals

   signal IR_out,data_bus1,lze_pc_out,lze_Amux_out,lze_Bmux_out,Amux_out, Bmux_out, registerA_out, registerB_out, registerMux_out, memory_out : std_logic_vector(31 downto 0);
   signal uze_imMUX1_out, imMUX1_out, lze_imMUX2_out, imMUX2_out, alu_out : std_logic_vector(31 downto 0);
   signal red_DataMem_out: unsigned (7 downto 0);
   signal zero_flag, carry_flag : std_logic;
   signal temp : std_logic_vector(30 downto 0) := (others => '0');
      begin
         ir_register : register32 port map (data_bus, ld_ir, clr_ir, clk, IR_out);
         a_register: register32 port map (Amux_out, ld_a, clr_a, clk, registerA_out);
         b_register: register32 port map(Bmux_out, ld_b, clr_b, clk, registerB_out);
         alu_component: alu port map(imMuX1_out, imMUX2_out, alu_op, alu_out, carry_flag, zero_flag);
         memory_component: memoryModule port map (mclk, RED_DataMem_out, registerMux_out, wen, en, memory_out);
         DATA_BUS <= data_bus1;
         lze_pc : lze port map (IR_out, lze_pc_out);
         lze_A_mux : lze port map (IR_out, lze_Amux_out);
         A_mux0 : mux2to1 port map (a_mux, data_bus, lze_Amux_out, Amux_out);
         lze_B_mux : lze port map (IR_out, lze_Bmux_out);
         B_mux0: mux2to1 port map (b_mux, data_bus, lze_Bmux_out, Bmux_out);
         reg_mux0: mux2to1 port map (reg_mux, registerA_out, registerB_out, registerMux_out);
         RED_DATA_MEM: red port map (IR_out, red_DataMem_out);
         UZE_IM_MUX1: uze port map (IR_out, UZE_imMUX1_out);
         IM_MUX1a : mux2to1 port map (in_mux1,registerA_out, UZE_imMUX1_out, imMUX1_out);
         LZE_IM_MUX2 : lze port map (IR_out, LZE_imMUX2_out);
         IM_MUX2a : mux4to1 port map (in_mux2, registerB_out, LZE_imMUX2_out, (temp & '1'), (others => '0'), imMUX2_out);
         DATA_MUX0: mux4to1 port map (data_mux, data_in, memory_out, alu_out, (others => '0'), data_bus1);
   end description;
```