library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity execute is

port( clk: in std\_logic;

execute\_fetch\_reg\_alu : out std\_logic\_vector(7 downto 0);

execute\_rlsmux\_sel: in std\_logic;

execute\_deop\_alu\_in: in std\_logic\_vector(3 downto 0);

execute\_dsel\_alu\_in:in std\_logic\_vector(3 downto 0);

execute\_decode\_mux2:in std\_logic\_vector(7 downto 0);

execute\_rst\_sig : in std\_logic;

execute\_rxen\_sig : in std\_logic;

execute\_rdx\_rx : in std\_logic\_vector(3 downto 0);

execute\_ryen\_sig : in std\_logic;

execute\_rdy\_ry : in std\_logic\_vector(3 downto 0);

execute\_writen\_sig: in std\_logic;

execute\_wrd\_waddr : in std\_logic\_vector(3 downto 0);

execute\_decode\_mux1:in std\_logic\_vector(7 downto 0);

execute\_ry\_sel : in std\_logic;

execute\_deop\_alu\_out: out std\_logic\_vector(3 downto 0);

execute\_reg\_alu : out std\_logic\_vector(7 downto 0);

execute\_mux2\_dmem : out std\_logic\_vector(7 downto 0);

execute\_alu\_mux3 : out std\_logic\_vector(7 downto 0);

execute\_mux3\_reg : in std\_logic\_vector(7 downto 0)

);

end execute;

architecture top of execute is

component regbank is

port( clk : in std\_logic;

rst : in std\_logic;

write\_en : in std\_logic;--\ enables for

rx\_en : in std\_logic;---> Write,Read rx 1 and ry 2.

ry\_en : in std\_logic;--/ comes from the decoder

rx\_addr : in std\_logic\_vector(3 downto 0);--Register addresses for Rx to be read

ry\_addr : in std\_logic\_vector(3 downto 0);--Register addresses for RY to be read

write\_addr : in std\_logic\_vector(3 downto 0);--Regbank address for where write\_data will be written

write\_data : in std\_logic\_vector(7 downto 0);--data being written into the regbank

rx\_data : out std\_logic\_vector(7 downto 0);--

ry\_data : out std\_logic\_vector(7 downto 0)

);

end component;

component mux is

port( a: in std\_logic\_vector(7 downto 0);

b: in std\_logic\_vector(7 downto 0);

sel: in std\_logic;

output: out std\_logic\_vector(7 downto 0)

) ;

end component mux;

component alu is

port( rx:in std\_logic\_vector(7 downto 0); -- input 1

ry: in std\_logic\_vector(7 downto 0); -- input 2

opcode: in std\_logic\_vector(3 downto 0); -- opcode in

sele: in std\_logic\_vector (3 downto 0); --select line

output: out std\_logic\_vector(7 downto 0) -- output

--branchen: out std\_logic-- branch enable line

--- zero: out std\_logic

);

end component alu;

signal reg\_alu: std\_logic\_vector(7 downto 0);

signal reg\_mux1: std\_logic\_vector(7 downto 0);

signal mux1\_alu: std\_logic\_vector(7 downto 0);

signal deop\_alu: std\_logic\_vector(3 downto 0);

begin

reg1: regbank port map(clk,execute\_rst\_sig,execute\_writen\_sig,execute\_rxen\_sig,execute\_ryen\_sig,execute\_rdx\_rx,execute\_rdy\_ry,execute\_wrd\_waddr,execute\_mux3\_reg,reg\_alu,reg\_mux1);

alu1: alu port map(reg\_alu,mux1\_alu,deop\_alu,execute\_dsel\_alu\_in,execute\_alu\_mux3);

mux1: mux port map(reg\_mux1,execute\_decode\_mux1,execute\_ry\_sel,mux1\_alu);

mux2: mux port map(execute\_decode\_mux2,reg\_mux1,execute\_rlsmux\_sel,execute\_mux2\_dmem);

execute\_reg\_alu <= reg\_alu;

execute\_fetch\_reg\_alu <= reg\_alu;

deop\_alu <= execute\_deop\_alu\_in;

execute\_deop\_alu\_out<= deop\_alu;

end architecture top;