library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity fetch is

port( clk: in std\_logic;

fetch\_reg\_alu: in std\_logic\_vector(7 downto 0); --

fetch\_deop\_alu: in std\_logic\_vector(3 downto 0);--

fetch\_offset\_sig: in std\_logic\_vector(7 downto 0);--

fetch\_pcjump\_sig: in std\_logic\_vector(7 downto 0);--

fetch\_instructions: out std\_logic\_vector(15 downto 0)--

);

end fetch;

architecture top of fetch is

component pcbranch is

port( rx\_in : in std\_logic\_vector(7 downto 0); -- connects to rx from regbank from regbank

opcode : in std\_logic\_vector(3 downto 0); --connects to alu\_opcode not select lines from decoder

offset : in std\_logic\_vector(7 downto 0); -- connects to branch from decoder

PC\_jump : in std\_logic\_vector(7 downto 0); --jump address for the PC from the Decoder.

clk : in std\_logic;

pcout : out std\_logic\_vector(7 downto 0) --outputs to instruction mem

);

end component;

component imem is

port( inst\_addr : in std\_logic\_vector(7 downto 0); -- input from PC

clk : in std\_logic;

instruction : out std\_logic\_vector(15 downto 0) -- output to go to decoder

);

end component imem;

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

begin

pcbranch1: pcbranch port map(fetch\_reg\_alu, fetch\_deop\_alu,fetch\_offset\_sig,fetch\_pcjump\_sig,clk,iaddress);

imem1: imem port map(iaddress,clk,fetch\_instructions);

end architecture top;