**Computer Hardware Experiments**

Lab\_08: FSM for multi-cycle processor

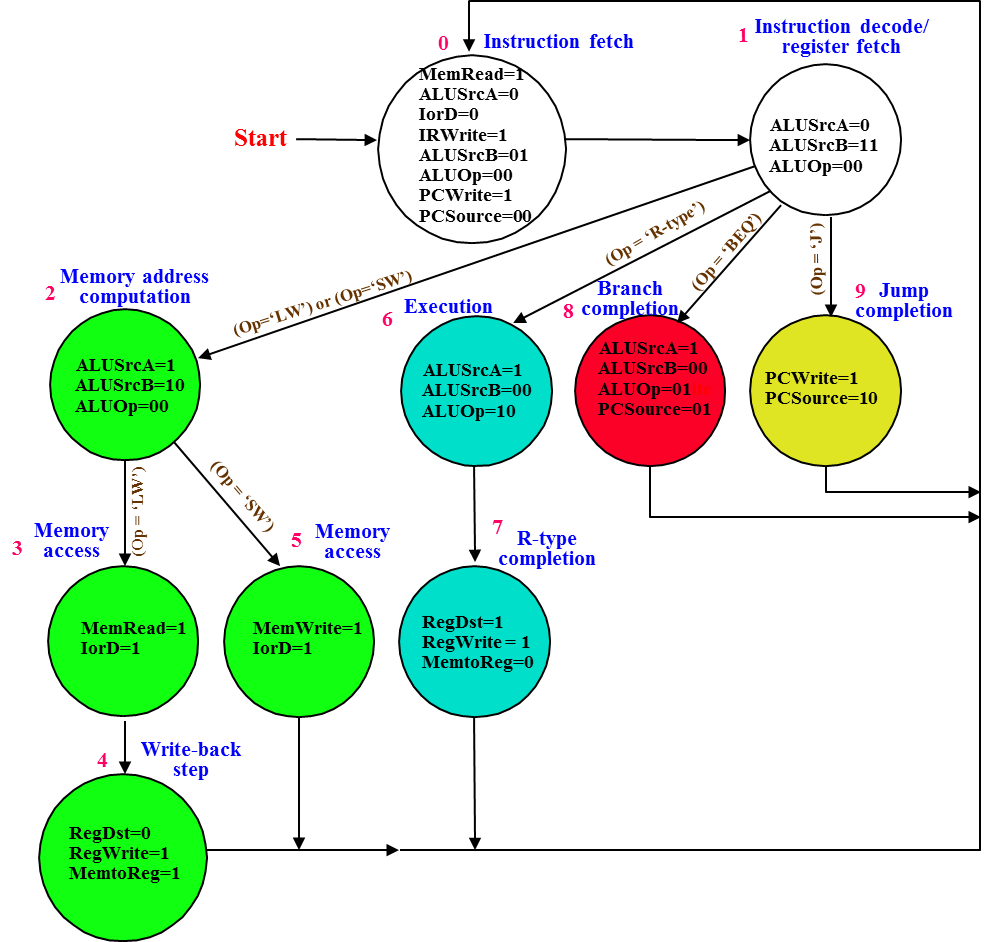
**Name : Student\_ID:**

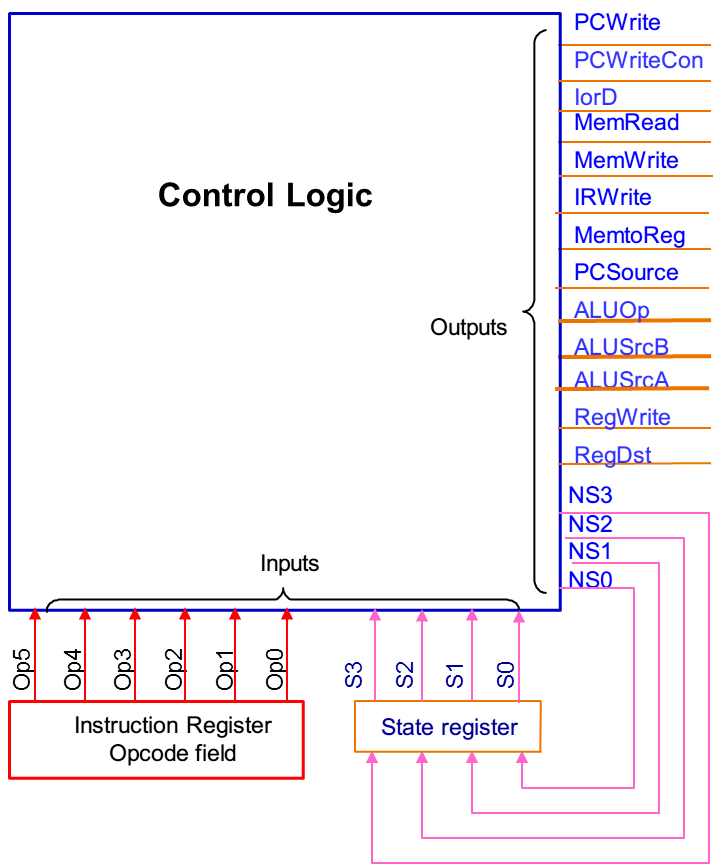
Purpose:

To understand and implement the function of FSM for multi-cycle processor.

You are going to implement an FSM that controls the operation of the multi-cycle processor. The following is the state transition diagram of the FSM:

Design a simple FSM as following:





In the state transition diagram, the state number of each state is given, you MUST use the number to represent each state in you VHDL. “*reset*” is a synchronous reset. When “*reset*” is 1, the FSM is at state 0. When “*reset*” is 0, the FSM transits following the state transition diagram. Note that the transition path is different for different *opcode*. Assume the opcode for each operation is as follows:

R-type: 000000

LW: 000001

SW: 000010

BEQ: 000011

J: 111111

In the state transition diagram, each state shows only the values of critical control signals, it is assumed that the value of a control signal is zero if it is not appeared in the diagram.

You MUST use the following entity to implement you FSM:

entity multicyclefsm is

port(

clk : in std\_logic;

reset : in std\_logic;

op : in std\_logic\_vector(5 downto 0);

pcwrite : out std\_logic;

pcwritecon: out std\_logic;

iord: out std\_logic;

memread: out std\_logic;

memwrite: out std\_logic;

irwrite: out std\_logic;

memtoreg: out std\_logic;

pcsource: out std\_logic\_vector(1 downto 0);

aluop: out std\_logic\_vector(1 downto 0);

alusrcb: out std\_logic\_vector(1 downto 0);

alusrca: out std\_logic;

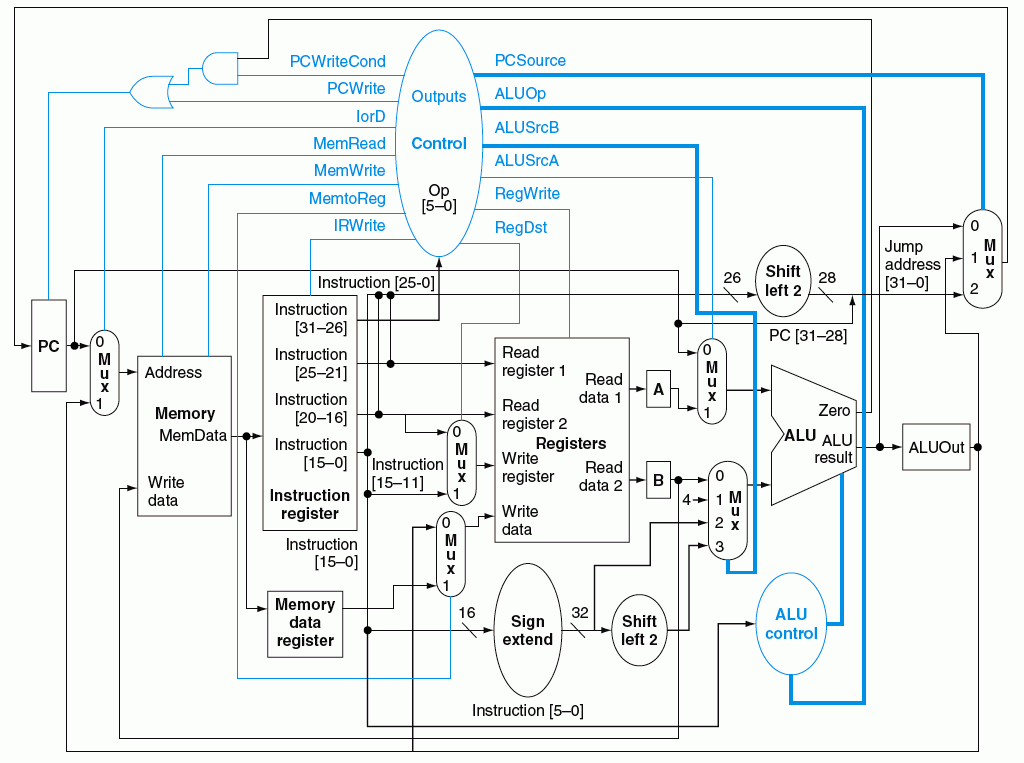
regwrite: out std\_logic;

regdst : out std\_logic;

statenum : out std\_logic\_vector(3 downto 0));

end multicyclefsm;

“statenum” is the current state number, you must assign the current state number to this output port.



You are going to use the test input as following:

stim\_proc: process

begin

wait for 2ns;

reset <= '1';

op <= "000000";

wait for 8ns;

reset <= '0';

op <= "000001";

wait for 50ns;

reset <= '1';

op <= "000000";

wait for 10ns;

reset <= '0';

op <= "000010";

wait for 40ns;

reset <= '1';

op <= "000000";

wait for 10ns;

reset <= '0';

op <= "000000";

wait for 40ns;

reset <= '1';

op <= "000000";

wait for 10ns;

reset <= '0';

op <= "000011";

wait for 30ns;

reset <= '1';

op <= "000000";

wait for 10ns;

reset <= '0';

op <= "111111";

wait for 30ns;

wait;

end process;