Lab Report No. 07:

Instruction Fetch

By:

Jiaqing Wan(16), Jiaxin Zheng(19)

Instructor: Dr. Xi Zhou

Classnumber: ZM1501

NAU ID:

5358836, 5358840

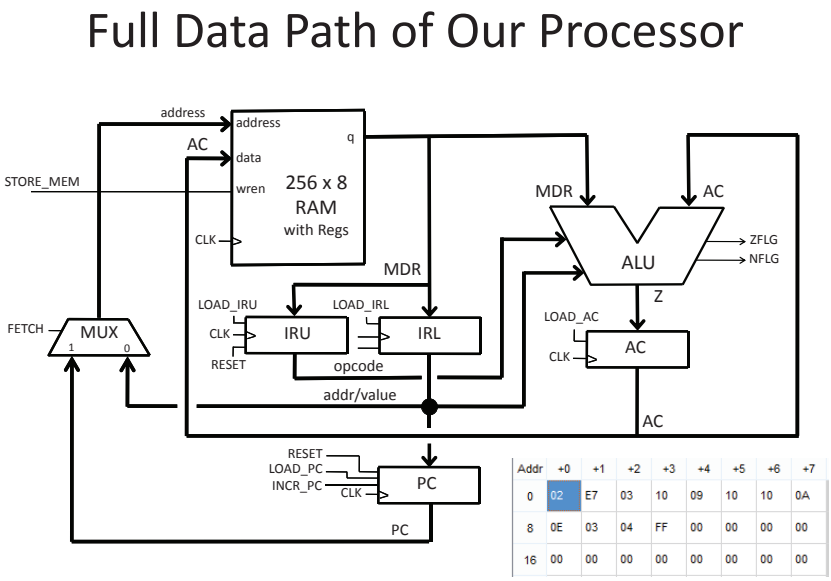
December 8, 2017

I. INTRODUCTION

A. Objectives

In this lab, the student will fully integrate the ALU, RAM, registers, and data paths of the microprocessor. We will also design the control unit needed to perform instruction fetches and manage the program counter. Testing will be by simulation on ModelSim.

B. Background knowledge



1. What the procedures to fetch and execute one instruction in our 8-bit microprocessor? Using one example to help explain it.

A:

First, fetch upper byte 0x02 whose address is 0x00 from RAM and storing itinto the IRU with incrementing PC. Then, fetch lower byte 0xE7 whose address is 0x01 from RAM and storing it into the IRL with incrementing PC.

Finally, execute LOADI which means load 0xE7 to AC.

2. What is state machine? How many states are there in the state machine built in this lab?
A:

State machine is composed of state register and combinational logic circuit.It can transfer state according to control signal and present state. It is a control center that coordinates related signal action and completes specific operation.

5 state machine in this lab.

3. Draw the State diagram of the state machine you built in this lab.

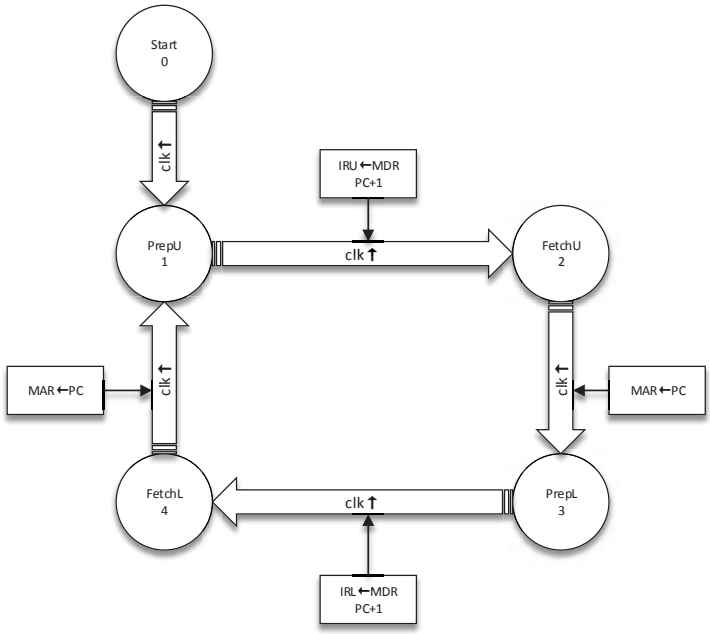


FIG. I.1 State diagram

C. Demonstration strategy

No demonstration in this lab.

D. Contribution of each team member

Jiaxin Zheng: Write code.

Jiaqing Wan: Do simulation and write report.

II. LAB DETAILS

A. VHDL code for state machine

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
   entity CU is port(
         opcode    :     in std_logic_vector (7 downto 0);
         NFLG      :     in std_logic ;
         ZFLG      :     in std_logic ;
         RESET     :     in std_logic ;
         CLK       :     in std_logic ;
                                              --define the input signal
         STATE     :     out std_logic_vector (2 downto 0);
         LOAD_AC   :     out std_logic ;
         LOAD_IRU  :     out std_logic ;
         LOAD_IRL  :     out std_logic ;
         LOAD_PC   :     out std_logic ;
         INCR_PC   :     out std_logic ;
         FETCH     :     out std_logic ;
         STORE_MEN:      out std_logic              --define the output signal
      );
   end CU;

   architecture behavioral of CU is
      type state_type is (start,prepu,fetchu,prep1,fetch1); --6 state_type
      signal present_state, next_state : state_type;
   begin

   --sync_proc:   -- synchronous process
   process (RESET, CLK)
   begin
      if RESET = '1' then
         present_state <= start;
      elsif (CLK'event and CLK = '0') then -- falling edge
         present_state <= next_state;
      end if;
   end process;
   --comb_proc:   -- combinational process
   process (present_state, next_state)
   begin
      case present_state is
         when start =>                --state 1  start
            STATE       <= "000";
            LOAD_AC     <= '0';
            LOAD_IRU    <= '0';
            LOAD_IRL    <= '0';
            LOAD_PC     <= '0';
            INCR_PC     <= '0';
            FETCH       <= '0';
            STORE_MEN   <= '0';
            next_state  <= prepu;
         when prepu =>               --state 2  prepare to load IRU
            STATE       <= "001";
            LOAD_AC     <= '0';
            LOAD_IRU    <= '0';
            LOAD_IRL    <= '0';
```

```
            LOAD_PC     <= '0';
            INCR_PC     <= '0';
            FETCH       <= '1';
            STORE_MEN   <= '0';
             next_state <= fetchu;
         when fetchu =>              --state 3  fetch
            STATE       <= "010";
            LOAD_AC     <= '0';
            LOAD_IRU    <= '1';
            LOAD_IRL    <= '0';
            LOAD_PC     <= '0';
            INCR_PC     <= '1';
            FETCH       <= '0';
            STORE_MEN   <= '0';
             next_state <= prep1;
         when prep1 =>               --state 4  prepare to load IRL
            STATE       <= "011";
            LOAD_AC     <= '0';
            LOAD_IRU    <= '0';
            LOAD_IRL    <= '0';
            LOAD_PC     <= '0';
            INCR_PC     <= '0';
            FETCH       <= '1';
            STORE_MEN   <= '0';
             next_state <= fetch1;
         when fetch1 =>              --state 5    fetch
            STATE       <= "100";
            LOAD_AC     <= '0';
            LOAD_IRU    <= '0';
            LOAD_IRL    <= '1';
            LOAD_PC     <= '0';
            INCR_PC     <= '1';
            FETCH       <= '0';
            STORE_MEN   <= '0';
             next_state <= prepu;
         when others => null;
      end case;
   end process;
   end behavioral;
```

B. Block diagram

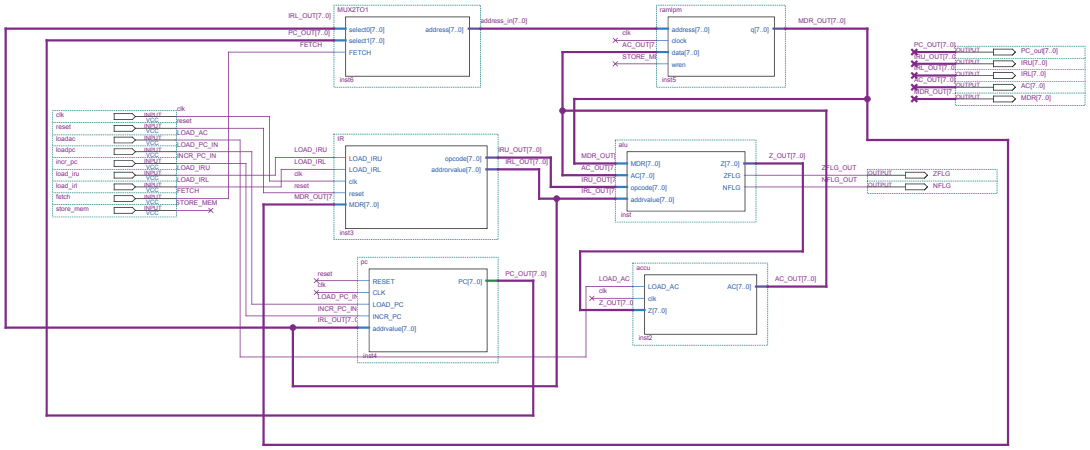


FIG. II.1 Block diagram without control unit

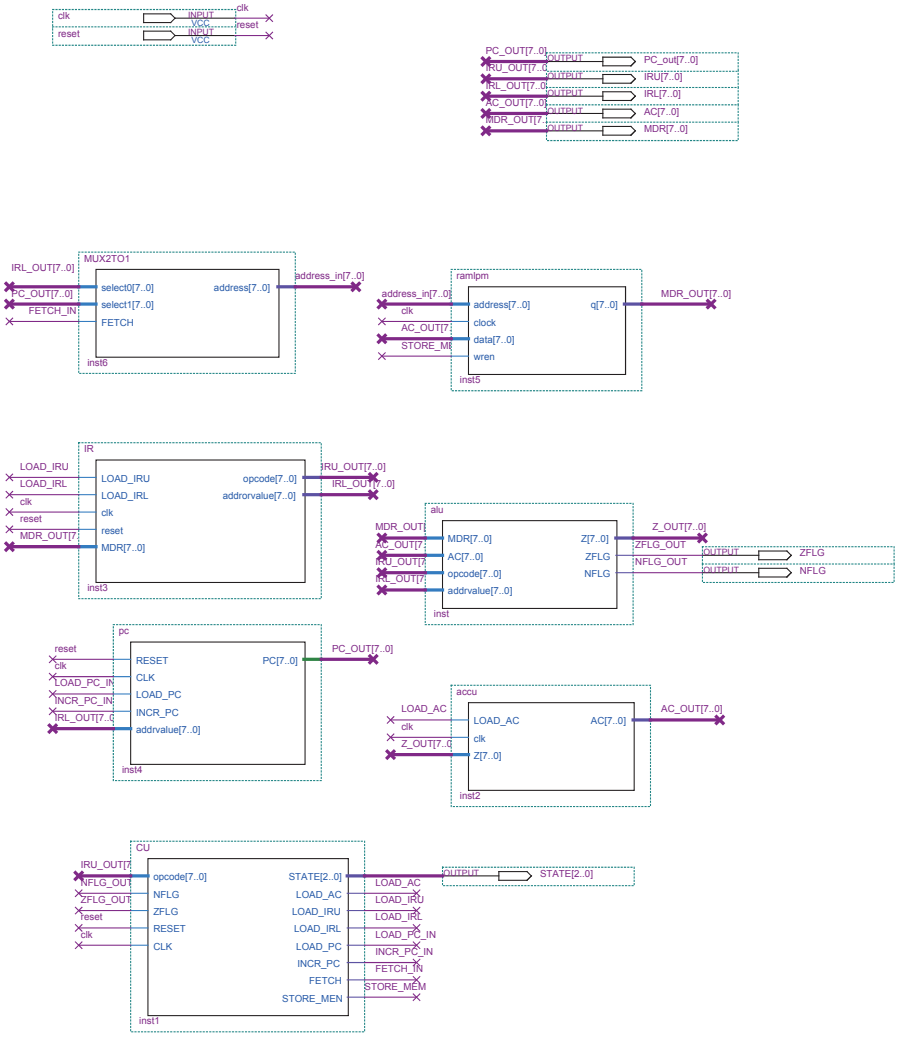


FIG. II.2 Block diagram with control unit

C. Simulation results and detailed explanation

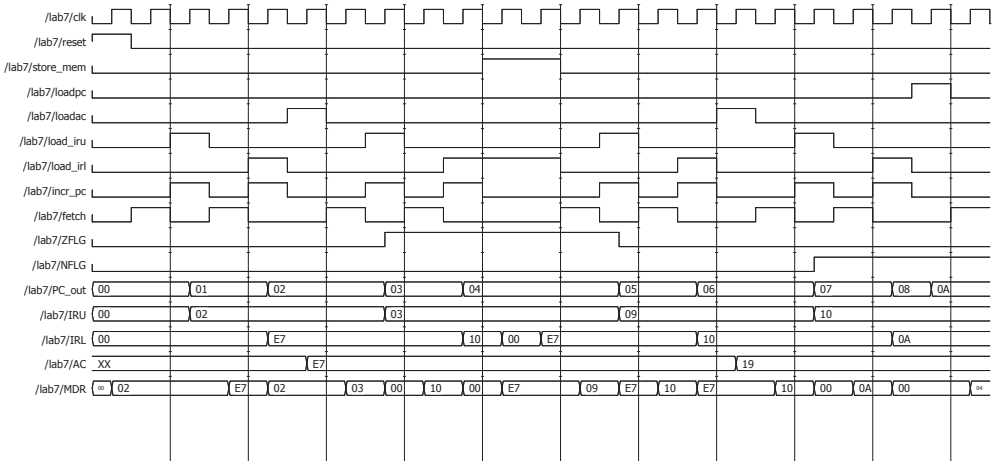


FIG. II.3 Wave

For first cycle,we should see like this table.

clk	1	2	3	4	5	6
reset	1	0	0	0	0	0
load ac	0	0	0	0	0	1
load pc	0	0	0	0	0	0
incr pc	0	0	1	0	1	0
load iru	0	0	1	0	0	0
load irl	0	0	0	0	1	0
fetch	0	1	0	1	0	0
store mem	0	0	0	0	0	0
PC	X"00"	X"00"	X"01"	X"01"	X"02"	X"02"
IRU(opcode)	X"00"	X"00"	X"02"	X"02"	X"02"	X"02"
IRL(address value)	X"00"	X"00"	X"00"	X"00"	E7	E7
AC	××	××	××	××	××	E7
MDR	02	02	02	E7	02	02

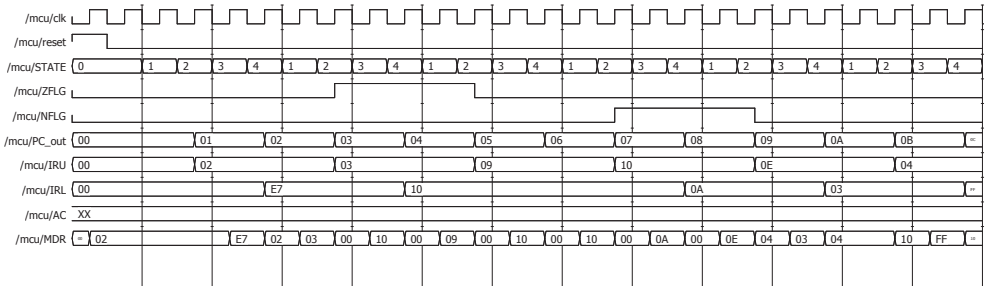


FIG. II.4 Wave

From the state diagram, we should observed like this table.

From the wave we can also observed the cycle of state 1,2,3,4.

State	0	1	2	3	4
reset	1	0	0	0	0
ZFLG	0	0	0	0	0
NFLG	0	0	0	0	0
PC	X"00"	X"00"	X"01"	X"01"	X"02"
IRU(opcode)	X"00"	X"00"	X"02"	X"02"	X"02"
IRL(address value)	X"00"	X"00"	X"00"	X"00"	E7
AC	××	××	××	××	××
MDR	02	02	02	E7	02

III. CONCLUSION

The lab is successful.

I realize all function we should achieve. But I meet some trouble on the force file instruction, I refer to manual and ask classmates that I solved it.