Pc_Unit

```
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity Pc is
33     Port ( input : in  STD_LOGIC_VECTOR (31 downto 0);
34            output : out  STD_LOGIC_VECTOR (31 downto 0);
35            CLK : in STD_LOGIC);
36 end Pc;
37
38 architecture Behavioral of Pc is
39 signal temp: STD_LOGIC_VECTOR(31 downto 0) := X"00000000";
40
41 begin
42 process (input ,CLK)
43 begin
44
45    if rising_edge(CLK) then
46       output <= temp;
47    end if;
48    if falling_edge(CLK) then
49       temp <= input;
50    end if;
51
52 end process;
53 end Behavioral;
54
55
```

Pc_Adder

```
30 --library UNISIM;
31 --use UNISIM.VComponents.all;
32
33 entity Pc_Adder is
34     Port ( i1 : in  STD_LOGIC_VECTOR (31 downto 0);
35            i2 : in  STD_LOGIC_VECTOR (31 downto 0);
36            output : out  STD_LOGIC_VECTOR (31 downto 0));
37 end Pc_Adder;
38
39 architecture Behavioral of Pc_Adder is
40
41 begin
42
43 output <= i1 + i2;
44
45 end Behavioral;
46
47
```

Branch_Adder

```
32
33  entity Adder is
34      Port ( ip1 : in  STD_LOGIC_VECTOR (31 downto 0);
35             ip2 : in  STD_LOGIC_VECTOR (31 downto 0);
36             op : out  STD_LOGIC_VECTOR (31 downto 0));
37  end Adder;
38
39  architecture Behavioral of Adder is
40
41  begin
42
43  op <= ip1 + ip2;
44
45  end Behavioral;
46
```

Instruction_Memory

```
31
32  entity Instruction_Memory is
33      Port ( pc : in  STD_LOGIC_VECTOR (31 downto 0);
34             instruction : out  STD_LOGIC_VECTOR (31 downto 0);
35             CLK: in STD_LOGIC
36             );
37  end Instruction_Memory;
38
39  architecture Behavioral of Instruction_Memory is
40
41  type arr is array(0 to 23) of STD_LOGIC_VECTOR (7 downto 0);
42
43  signal memory: arr:= (
44
45  "00000000", "10000101", "00010000", "00100000", -- add $v0, $a0, $a1
46  "10101100", "00000010", "00000000", "00001000", -- sw $v0, 8($Zero)
47  "10001100", "00000110", "00000000", "00001000", -- lw $a2, 8($Zero)
48  "00010000", "11000010", "00000000", "00000001", -- beq $v0, $a2, 1
49  "00000000", "01000110", "10001000", "00101010", -- slt $s1, $v0, $a2
50  "00000000", "10100100", "10001000", "00100010"  -- sub $s1, $a1, $a0
51  );
52
53  begin
54
55
56        instruction(31 downto 24) <= memory(to_integer(unsigned(pc)));
57        instruction(23 downto 16) <= memory(to_integer(unsigned(pc)+1));
58        instruction(15 downto 8) <= memory(to_integer(unsigned(pc)+2));
59        instruction(7 downto 0) <= memory(to_integer(unsigned(pc)+3));
60
61
62  end Behavioral;
63
```

Control_Unit

```
entity control is
    Port ( Operation : in  STD_LOGIC_VECTOR (5 downto 0);
           RegDst : out  STD_LOGIC;
           AluSrc : out  STD_LOGIC;
           Memoryreg : out  STD_LOGIC;
           RegWrite : out  STD_LOGIC;
           MemRead : out  STD_LOGIC;
           MemWrite : out  STD_LOGIC;
           Branch : out  STD_LOGIC;
           AluOp : out  STD_LOGIC_VECTOR (1 downto 0));
end control;

architecture Behavioral of control is

begin

process(Operation)
begin

   if Operation = "000000" then
           RegDst <= '1';
           AluSrc <= '0';
           Memoryreg <= '0';
           RegWrite <= '1';
           MemRead <= '0';
           MemWrite <= '0';
           Branch <= '0';
           AluOp <= "10";
   elsif Operation = "100011"then
           RegDst <= '0';
           AluSrc <= '1';
           Memoryreg <= '1';
           RegWrite <= '1';
           MemRead <= '1';
           MemWrite <= '0';
           Branch <= '0';
           AluOp <= "00";
   elsif Operation = "101011" then
           RegDst <= '1';
           AluSrc <= '1';
           Memoryreg <= '1';
           RegWrite <= '0';
           MemRead <= '0';
           MemWrite <= '1';
           Branch <= '0';
           AluOp <= "00";
   elsif Operation <= "000100" then
           RegDst <= '1';
           AluSrc <= '0';
           Memoryreg <= '1';
           RegWrite <= '0';
           MemRead <= '0';
           MemWrite <= '0';
           Branch <= '1';
           AluOp <= "01";
           end if;
end process;
```

Mux_RegDst

```
entity MemReg_Mux is
    Port ( i1 : in  STD_LOGIC_VECTOR (4 downto 0);
           i2 : in  STD_LOGIC_VECTOR (4 downto 0);
           sel_line : in  STD_LOGIC;
           output : out  STD_LOGIC_VECTOR (4 downto 0));
end MemReg_Mux;

architecture Behavioral of MemReg_Mux is

begin
process (i1,i2,sel_line)
begin

   if sel_line = '0' then
      output <= i1;
   else
      output <= i2;
   end if;

end process;
end Behavioral;
```

Mux_Memory

```
31
32  entity Memory_Mux is
33      Port ( i1 : in  STD_LOGIC_VECTOR (31 downto 0);
34             i2 : in  STD_LOGIC_VECTOR (31 downto 0);
35             sel_line : in  STD_LOGIC;
36             output : out  STD_LOGIC_VECTOR (31 downto 0));
37  end Memory_Mux;
38
39  architecture Behavioral of Memory_Mux is
40
41  begin
42  process (i1,i2,sel_line)
43  begin
44     if sel_line <= '0' then
45        output <= i1;
46     else
47        output <= i2;
48     end if;
49  end process;
50  end Behavioral;
51
```

Mux_Alu

```
1
2   entity Alu_Mux is
3       Port ( i1 : in  STD_LOGIC_VECTOR (31 downto 0);
4              i2 : in  STD_LOGIC_VECTOR (31 downto 0);
5              sel_line : in  STD_LOGIC;
6              output : out  STD_LOGIC_VECTOR (31 downto 0));
7   end Alu_Mux;
8
9   architecture Behavioral of Alu_Mux is
0
1   begin
2   process (i1,i2,sel_line)
3   begin
4      if sel_line = '0' then
5         output <= i1;
6      else
7         output <= i2;
8         end if;
9   end process;
0   end Behavioral;
```

Mux_Pc

```
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity Pc_Mux is
33      Port ( i1 : in  STD_LOGIC_VECTOR (31 downto 0);
34             i2 : in  STD_LOGIC_VECTOR (31 downto 0);
35             sel_line : in  STD_LOGIC;
36             output : out  STD_LOGIC_VECTOR (31 downto 0));
37  end Pc_Mux;
38
39  architecture Behavioral of Pc_Mux is
40
41  begin
42  process (i1 , i2 ,sel_line)
43  begin
44     if(sel_line = '0') then
45        output <= i1;
46     elsif(sel_line = '1') then
47        output <= i2;
48
49  end if;
50  end process;
51  end Behavioral;
52
```

Register_File

```
31
32  entity RegFile is
33      Port ( read_Reg1 : in  STD_LOGIC_VECTOR (4 downto 0);
34             read_Reg2 : in  STD_LOGIC_VECTOR (4 downto 0);
35             write_Reg : in  STD_LOGIC_VECTOR (4 downto 0);
36             write_data : in  STD_LOGIC_VECTOR (31 downto 0);
37             read_data1 : out  STD_LOGIC_VECTOR (31 downto 0);
38             read_data2 : out  STD_LOGIC_VECTOR (31 downto 0);
39             CLK: in STD_LOGIC;
40             RegWrite : in  STD_LOGIC);
41  end RegFile;
42
43  architecture Behavioral of RegFile is
44
45  type RegFileType is array (0 to 31) of STD_LOGIC_VECTOR (31 downto 0);
46
47  signal array_reg : RegFileType := ( X"00000000", X"00000000", X"00000000", X"00000000",
48                                      X"00000005", X"00000007", X"00000000", X"00000000",
49                                      X"00000000", X"00000000", X"00000000", X"00000000",
50                                      X"00000000", X"00000000", X"00000000", X"00000000",
51                                      X"00000000", X"00000000", X"00000000", X"00000000",
52                                      X"00000000", X"00000000", X"00000000", X"00000000",
53                                      X"00000000", X"00000000", X"00000000", X"00000000",
54                                      X"00000000", X"00000000", X"00000000", X"00000000");
55
56
57  begin
58  read_data1 <= array_reg(to_integer(unsigned(read_Reg1)));
59  read_data2 <= array_reg(to_integer(unsigned(read_Reg2)));
60
61  process(RegWrite,CLK)
62  begin
63     if (RegWrite = '1' and rising_edge(CLK)) then
64        array_reg(to_integer(unsigned(write_Reg))) <= write_data;
65
66     end if;
67  end process;
68  end Behavioral;
69
```

Sign_Extend

```
entity Sign_Extend is
    Port ( input : in  STD_LOGIC_VECTOR (15 downto 0);
           output : out  STD_LOGIC_VECTOR (31 downto 0));
end Sign_Extend;

architecture Behavioral of Sign_Extend is

begin

process (input)
begin

output(15 downto 0) <= input(15 downto 0);

if input(15) = '0' then
   output(31 downto 16) <= X"0000";

elsif input(15) = '1' then
   output(31 downto 16) <= X"FFFF";



end if;
end process;

end Behavioral;
```

Shift_Left_2

```
entity shift_left_32_32 is
    Port ( input : in  STD_LOGIC_VECTOR (31 downto 0);
           output : out  STD_LOGIC_VECTOR (31 downto 0));
end shift_left_32_32;

architecture Behavioral of shift_left_32_32 is

begin

output(31 downto 2) <= input(29 downto 0);
output(1 downto 0) <= "00";

end Behavioral;
```

Alu_Control

```
entity Alu_Control is
    Port ( AluOp : in  STD_LOGIC_VECTOR (1 downto 0);
           Instruction : in  STD_LOGIC_VECTOR (5 downto 0);
           operation : out  STD_LOGIC_VECTOR (3 downto 0));
end Alu_Control;

architecture Behavioral of Alu_Control is

begin

process(AluOp,Instruction)
begin

if(AluOp = "00" )then
   operation <= "0010";

elsif AluOp = "01" then
   operation <= "0110";

elsif AluOp = "10" then
   if Instruction(3 downto 0) = "0000" then
      operation <= "0010";
   elsif Instruction(3 downto 0) = "0100" then
      operation <= "0000";
   elsif Instruction(3 downto 0) = "0101" then
      operation <= "0001";
```

```
   elsif AluOP(1) = '1' then
      if Instruction(3 downto 0) = "0010" then
      operation <= "0110";
      elsif Instruction(3 downto 0) = "1010" then
      operation <= "0111";
      end if;
   end if;
elsif AluOP(1) = '1' then
      if Instruction(3 downto 0) = "0010" then
      operation <= "0110";
      elsif Instruction(3 downto 0) = "1010" then
      operation <= "0111";
   end if;
 end if;

end process;
```

Alu_Unit

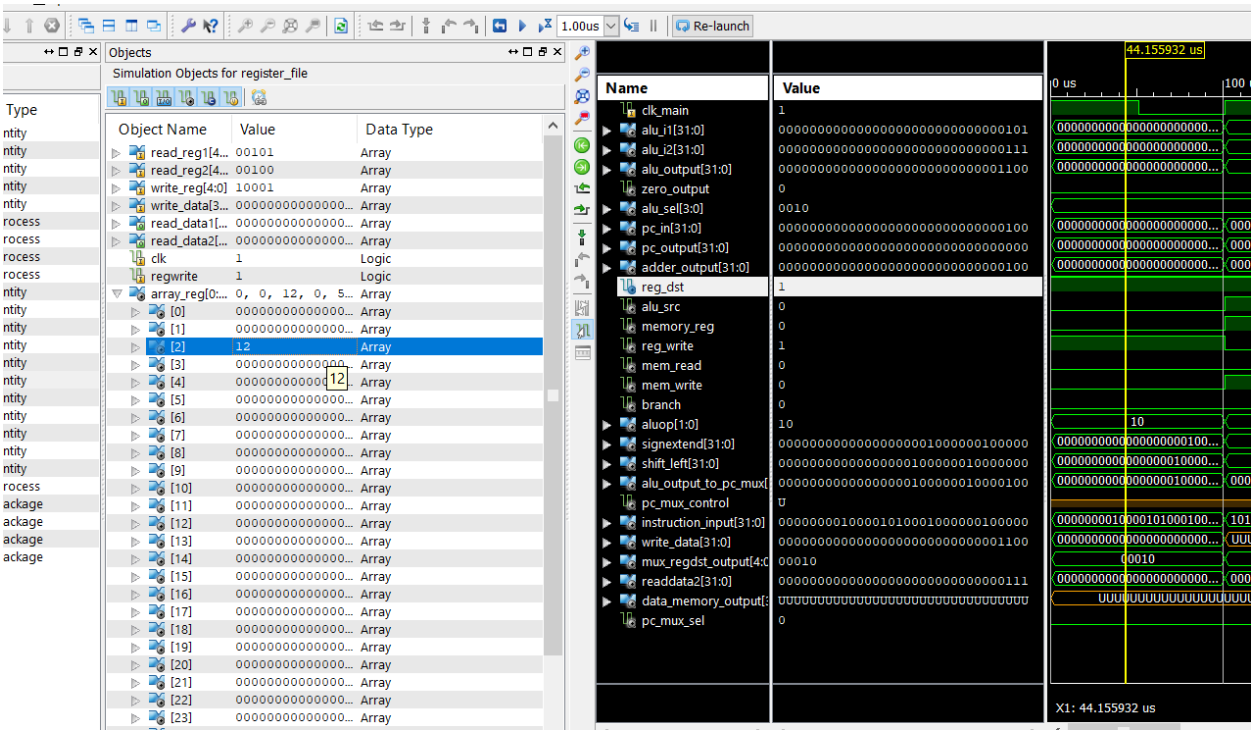
```
entity ALU is
    Port ( A : in  STD_LOGIC_VECTOR (31 downto 0);
           B : in  STD_LOGIC_VECTOR (31 downto 0);
           AluOperation : in  STD_LOGIC_VECTOR (3 downto 0);
           Zero : out  STD_LOGIC;
           res : out  STD_LOGIC_VECTOR (31 downto 0));
end ALU;

architecture Behavioral of ALU is

begin
process (A,B,AluOperation)
begin
   if AluOperation = "0000"  then
         res <= A AND B;
   elsif AluOperation = "0001" then
         res <= A OR B;
   elsif AluOperation = "0111" then
         if A < B then
         res <= X"00000001";
         else  res <= X"00000000";
         end if;
         end if;
   elsif AluOperation = "0010" then
         res <= A + B;
   elsif AluOperation = "0110" then
         res <= A - B;
   elsif AluOperation = "1100" then
         res <= A NOR B;
   end if;
if A = B then
   Zero <= '1';
else
   Zero <= '0';
end if;
end process;
end Behavioral;
```

Data_Memory

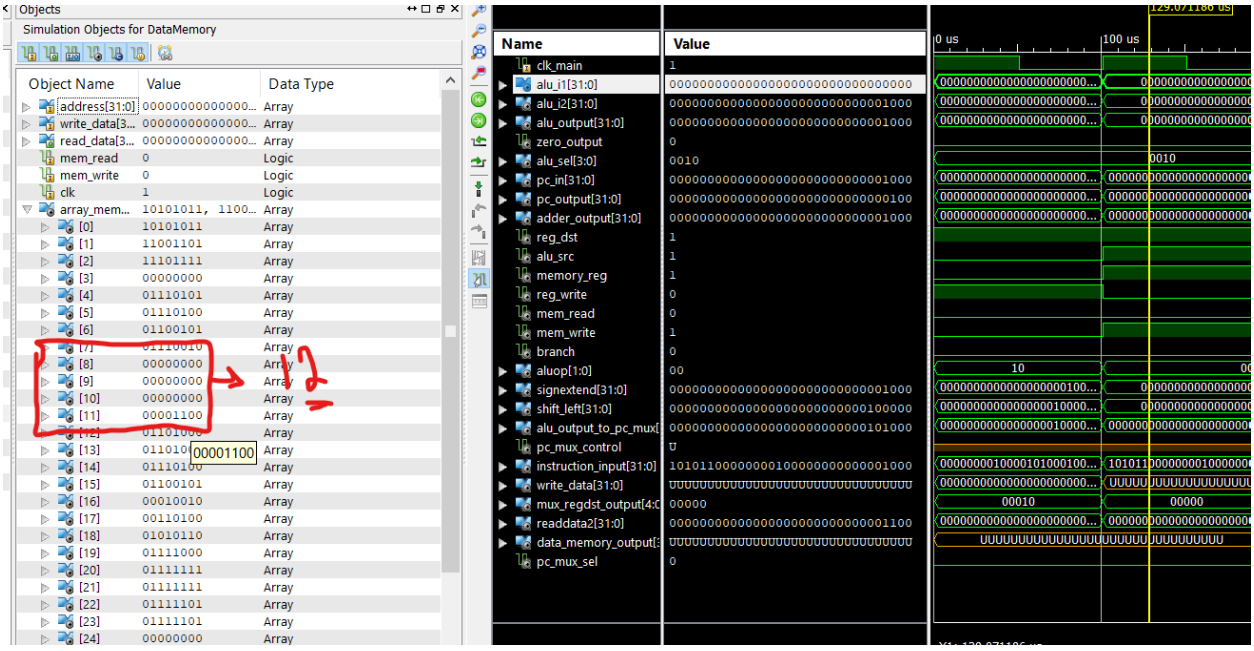
```
31
32  entity Data_Memory is
33      Port ( Address : in  STD_LOGIC_VECTOR (31 downto 0);
34             write_Data : in  STD_LOGIC_VECTOR (31 downto 0);
35             read_Data : out  STD_LOGIC_VECTOR (31 downto 0);
36             mem_read : in  STD_LOGIC;
37             mem_write : in  STD_LOGIC;
38             CLK: in STD_LOGIC);
39  end Data_Memory;
40
41    architecture Behavioral of Data_Memory is
42
43  type Data_Memory_Type is array (0 to 35) of STD_LOGIC_VECTOR (7 downto 0);
44
45  signal array_Memory : Data_Memory_Type := (
46                                      X"AB", X"CD", X"EF", X"00",
47                                      X"75", X"74", X"65", X"72",
48                                      X"20", X"41", X"72", X"63",
49                                      X"68", X"69", X"74", X"65",
50                                      X"12", X"34", X"56", X"78",
51                                      X"7F", X"7F", X"7D", X"7D",
52                                      X"00", X"00", X"00", X"00",
53                                      X"78", X"78", X"6A", X"6A",
54                                      X"00", X"00", X"00", X"01");
55  begin
56
54                                      X"00", X"00", X"00", X"01");
55  begin
56
57  process(Address, mem_read, mem_write,CLK)
58  begin
59     if(mem_read = '1' and mem_write = '0' ) then
60        read_Data(31 downto 24) <= array_Memory(to_integer(unsigned(Address)));
61        read_Data(23 downto 16) <= array_Memory(to_integer(unsigned(Address)+1));
62        read_Data(15 downto 8) <= array_Memory(to_integer(unsigned(Address)+2));
63        read_Data(7 downto 0) <= array_Memory(to_integer(unsigned(Address)+3));
64
65
66     elsif(mem_read = '0' and mem_write = '1' and rising_edge(CLK) ) then
67        array_Memory(to_integer(unsigned(Address))) <= write_Data(31 downto 24);
68        array_Memory(to_integer(unsigned(Address)+1)) <= write_Data(23 downto 16);
69        array_Memory(to_integer(unsigned(Address)+2)) <= write_Data(15 downto 8);
70        array_Memory(to_integer(unsigned(Address)+3)) <= write_Data(7 downto 0);
71     end if;
72
73  end process;
74  end Behavioral;
75
```

ADD $v0, $a0, $a1

$v0 = 12

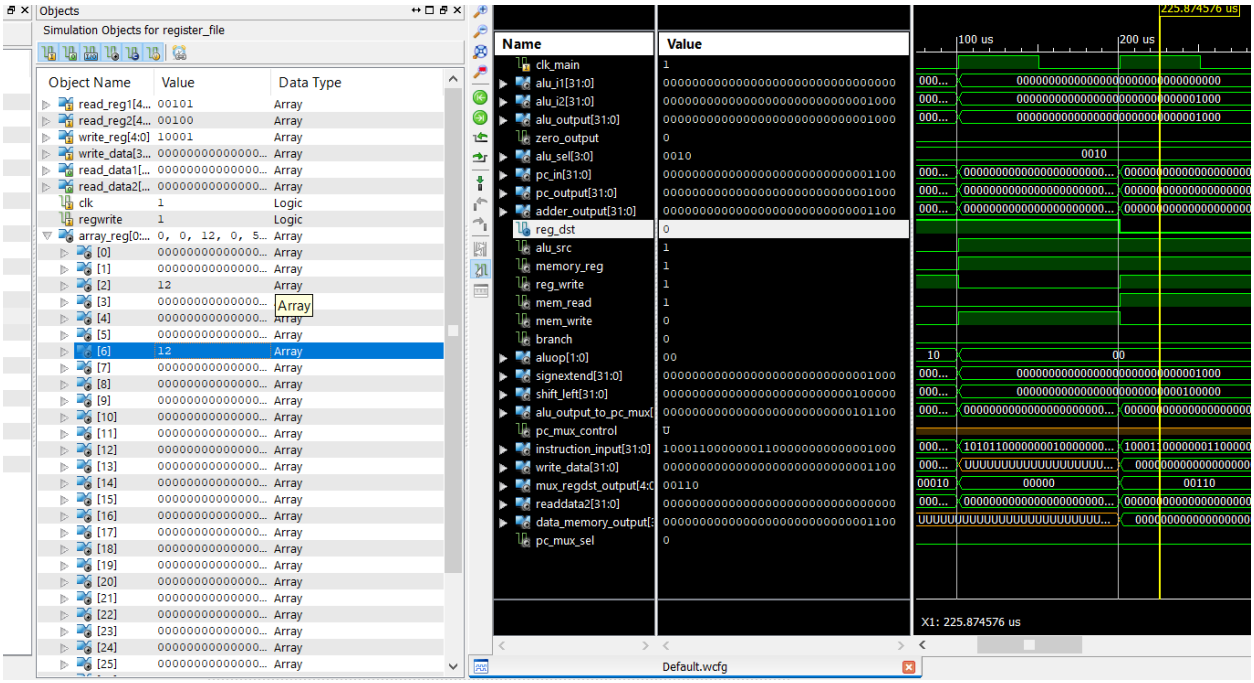


SW $V0 , 8($ZERO)

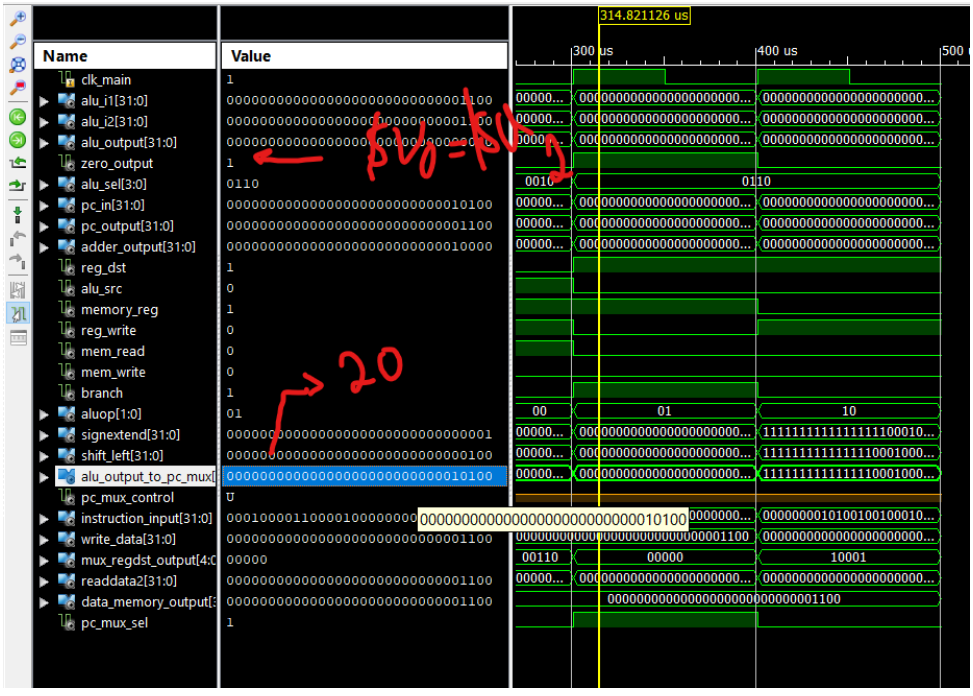


LW $a2, $8($Zero)

$a2 = 12



Beq $v0 , $a2, Good_Processor



Sub $s1 , $a1, $a0

$a1 - $a0 = 7 – 5 = 2

$s1 = 2