Ty Madsen
ECEN 220
20 November 2013

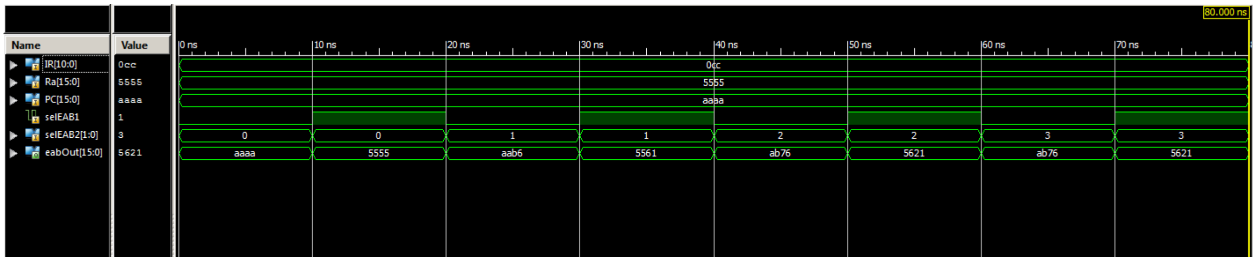
Lab 10 - LC-3 Functional Units

EAB:

Verilog:

```
module EAB(
   input [10:0] IR,
   input [15:0] Ra,
   input [15:0] PC,
   input selEAB1,
   input [1:0] selEAB2,
   output [15:0] eabOut
   );

         wire [15:0] opA, opB;

         assign opA = (selEAB2 == 2'b00) ? (16'b0000000000000000) :
                                                     (selEAB2 == 2'b01) ? {{10{IR[5]}} , IR[5:0]} :
                                                     (selEAB2 == 2'b10) ? {{7{IR[8]}} , IR[8:0]} :

        {{6{IR[10]}} , IR[10:0]};

         assign opB = selEAB1 == 1'b0 ? PC : Ra;

         assign eabOut = (opA + opB);

endmodule
```

Simulation:

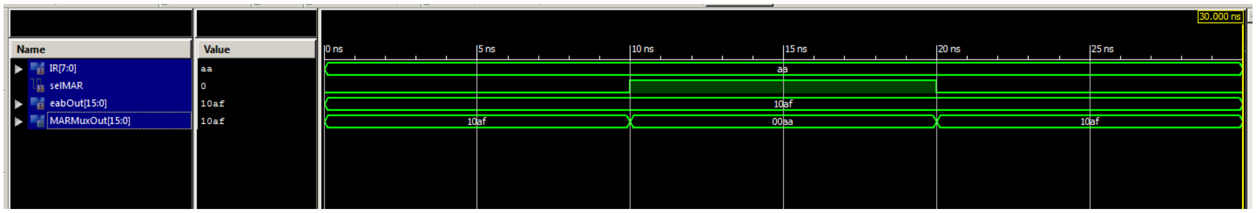


MARMux:

Verilog:

```
module MARMux(
   input [7:0] IR,
   input selMAR,
   input [15:0] eabOut,
   output [15:0] MARMuxOut
   );

		assign MARMuxOut = selMAR == 1'b0 ? eabOut : {{8{1'b0}}, IR};


endmodule
```

Simulation:



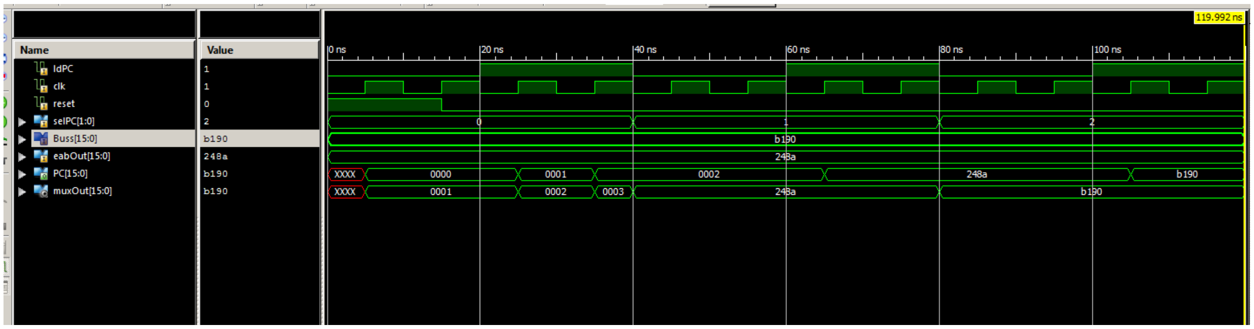
| Name | Value |
|---|---|
| IR[7:0] | aa |
| selMAR | 0 |
| eabOut[15:0] | 10af |
| MARMuxOut[15:0] | 10af |

PC:

Verilog:

```
module PC(
   input ldPC,
   input clk,
   input reset,
   input [1:0] selPC,
   input [15:0] Buss,
   input [15:0] eabOut,
   output [15:0] PC
   );

            wire [15:0] muxOut;

            assign muxOut = selPC == 2'b00 ? (PC + {{15{1'b0}}, 1'b1}) :
                                                                  selPC == 2'b01 ? eabOut :
                                                                  selPC == 2'b10 ? Buss :
                                                                  PC;

            register PCReg(PC, clk, muxOut, reset, ldPC);


endmodule
```

Simulation:



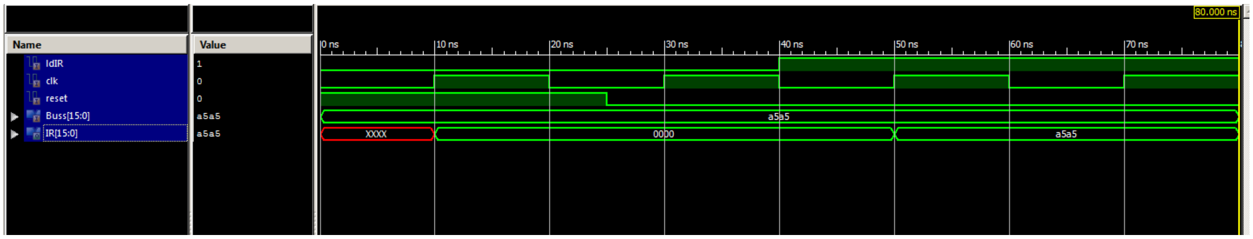
| Name | Value |
|---|---|
| ldPC | 1 |
| clk | 1 |
| reset | 0 |
| selPC[1:0] | 2 |
| Buss[15:0] | b190 |
| eabOut[15:0] | 248a |
| PC[15:0] | b190 |
| muxOut[15:0] | b190 |

IR:

Verilog:

```
module IR(
   input ldIR,
   input clk,
   input reset,
   input [15:0] Buss,
   output [15:0] IR
   );

            register IRReg(IR, clk, Buss, reset, ldIR);

endmodule
```

Simulation:

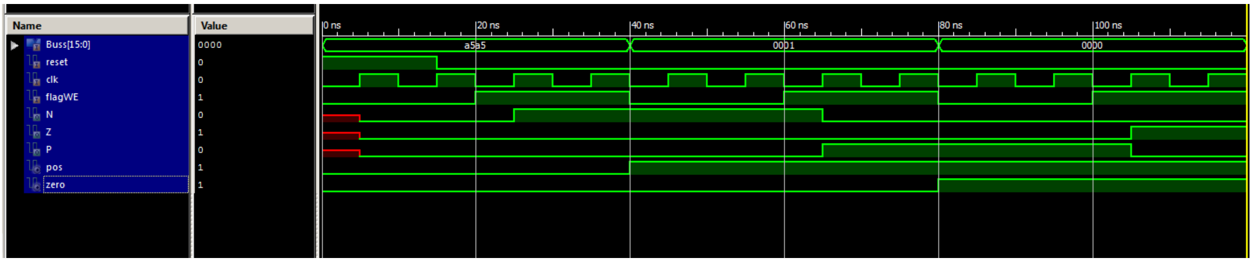


NZP:

Verilog:

```
module NZP(
    input [15:0] Buss,
    input reset,
    input clk,
    input flagWE,
    output N,
    output Z,
    output P
    );

        wire pos, zero;

        assign pos = ~Buss[15];
        assign zero = &(~Buss);

       ff_dce ff1(N, clk, ~pos, reset, flagWE);
       ff_dce ff2(Z, clk, zero, reset, flagWE);
       ff_dce ff3(P, clk, pos&~zero, reset, flagWE);

endmodule
```

Simulation:



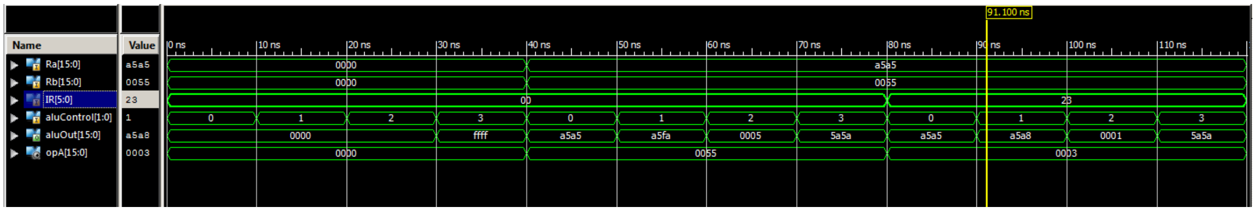
| Name | Value |
|---|---|
| Buss[15:0] | 0000 |
| reset | 0 |
| clk | 0 |
| flagWE | 1 |
| N | 0 |
| Z | 1 |
| P | 0 |
| pos | 1 |
| zero | 1 |

ALU:

Verilog:

```
module ALU(
   input [15:0] Ra,
   input [15:0] Rb,
   input [5:0] IR,
   input [1:0] aluControl,
   output [15:0] aluOut
   );

            wire [15:0] opA;

            assign opA = (IR[5] == 1'b0) ? Rb : {{11{IR[4]}}, IR[4:0]};

            assign aluOut =           aluControl == 2'b00 ? Ra :
                                                          aluControl == 2'b01 ? (Ra + opA) :
                                                          aluControl == 2'b10 ? (Ra & opA) :
                                                          ~Ra;
endmodule
```

Simulation:



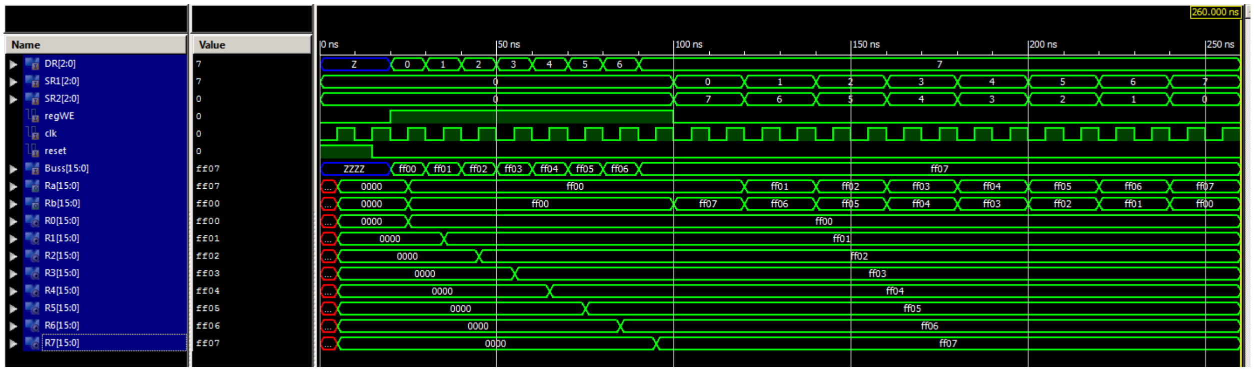
RegFile:

Verilog:

```
module RegFile(
  input [2:0] DR,
  input [2:0] SR1,
  input [2:0] SR2,
  input regWE,
  input clk,
  input reset,
  input [15:0] Buss,
  output [15:0] Ra,
  output [15:0] Rb
  );

        wire [15:0] R0, R1, R2, R3, R4, R5, R6, R7;

        register reg0(R0, clk, Buss, reset, (regWE & DR == 3'b000));
        register reg1(R1, clk, Buss, reset, (regWE & DR == 3'b001));
        register reg2(R2, clk, Buss, reset, (regWE & DR == 3'b010));
        register reg3(R3, clk, Buss, reset, (regWE & DR == 3'b011));
        register reg4(R4, clk, Buss, reset, (regWE & DR == 3'b100));
        register reg5(R5, clk, Buss, reset, (regWE & DR == 3'b101));
        register reg6(R6, clk, Buss, reset, (regWE & DR == 3'b110));
        register reg7(R7, clk, Buss, reset, (regWE & DR == 3'b111));

        assign Ra =     SR1 == 3'b000 ? R0 :
                                              SR1 == 3'b001 ? R1 :
                                              SR1 == 3'b010 ? R2 :
                                              SR1 == 3'b011 ? R3 :
                                              SR1 == 3'b100 ? R4 :
                                              SR1 == 3'b101 ? R5 :
                                              SR1 == 3'b110 ? R6 :
                                              R7;
        assign Rb =     SR2 == 3'b000 ? R0 :
                                              SR2 == 3'b001 ? R1 :
                                              SR2 == 3'b010 ? R2 :
                                              SR2 == 3'b011 ? R3 :
                                              SR2 == 3'b100 ? R4 :
                                              SR2 == 3'b101 ? R5 :
                                              SR2 == 3'b110 ? R6 :
                                              R7;
```

```
endmodule
```

Simulation:



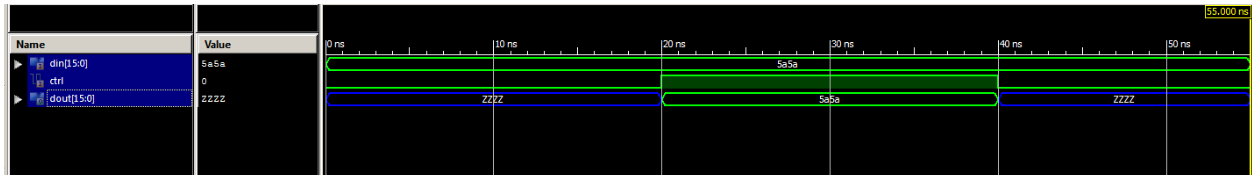
| Name | Value |
|---|---|
| DR[2:0] | 7 |
| SR1[2:0] | 7 |
| SR2[2:0] | 0 |
| regWE | 0 |
| clk | 0 |
| reset | 0 |
| Buss[15:0] | ff07 |
| Ra[15:0] | ff07 |
| Rb[15:0] | ff00 |
| R0[15:0] | ff00 |
| R1[15:0] | ff01 |
| R2[15:0] | ff02 |
| R3[15:0] | ff03 |
| R4[15:0] | ff04 |
| R5[15:0] | ff05 |
| R6[15:0] | ff06 |
| R7[15:0] | ff07 |

ts_driver:

Verilog:

```
module ts_driver ( din, dout, ctrl );
            input [15:0] din;
           input ctrl;
           output [15:0] dout;

           assign dout = (ctrl)? din:(16'bZZZZZZZZZZZZZZZZ);
endmodule
```

Simulation:


55.000 ns
Name
Value
din[15:0]
5a5a
ctrl
0
dout[15:0]
zzzz
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
5a5a
ZZZZ
5a5a
ZZZZ