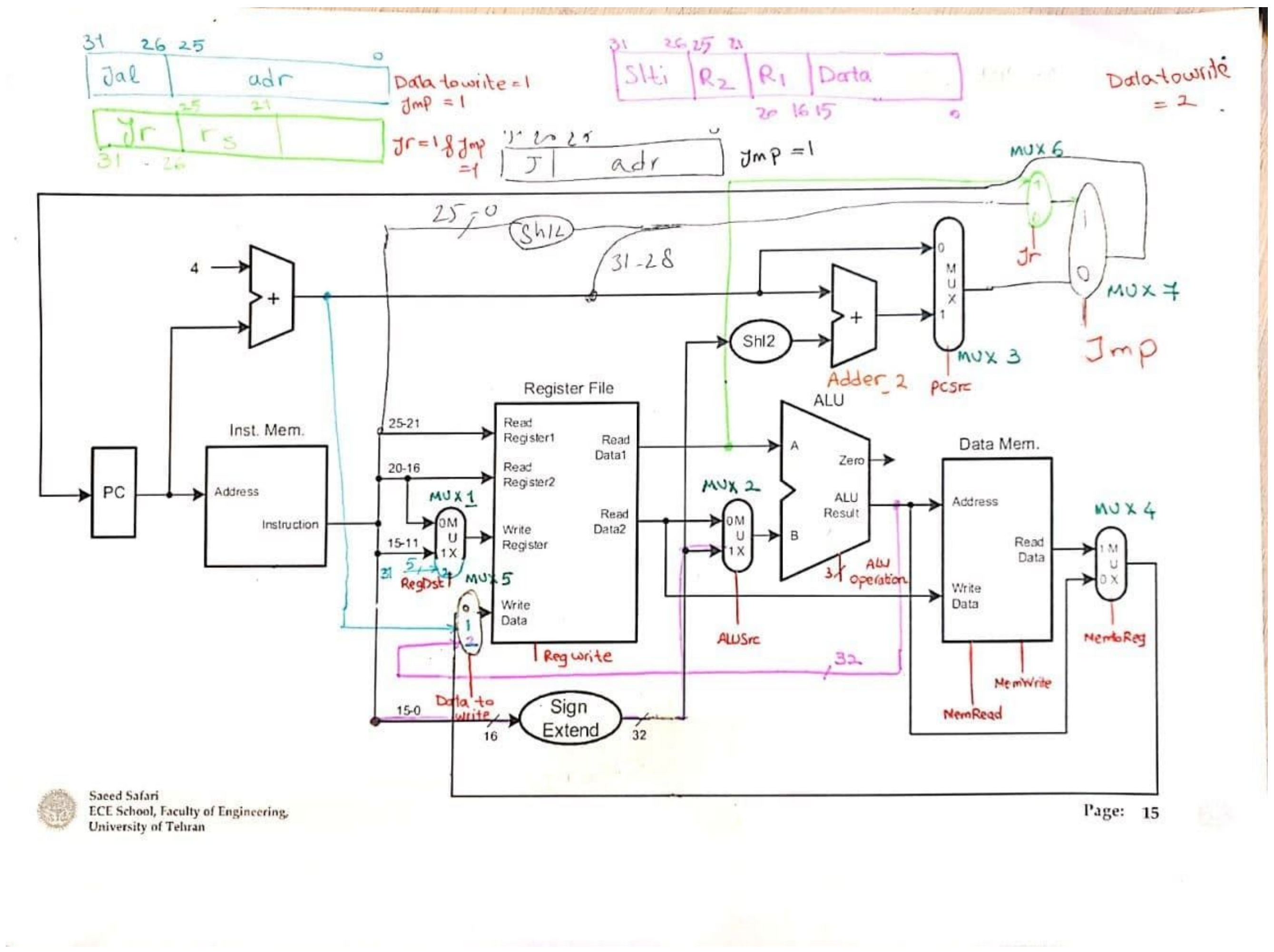
Computer Assignment #2 (Single Cycle MIPS Processor)

Helia Hoseini (810197491)
Daneshvar Amrollahi (810197685)

Datapath:

Block Diagram:



Verilog Code

```
 reg_32b PC(mux7_out, rst, 1'b1, clk, pc_out);

 adder_32b ADDER_1 (pc_out , 32'd4, 1'b0, , adder1_out);

 mux3to1_5b MUX_1(inst[20:16], inst[15:11], 5'b11111, reg_dst, mux1_out);

 mux3to1_32b MUX_5(mux4_out,  adder1_out, alu_out, data_to_write, mux5_out);

 reg_file  RF(mux5_out, inst[25:21], inst[20:16], mux1_out, reg_write, rst,
clk, read_data1, read_data2);

 sign_ext SGN_EXT(inst[15:0], sgn_ext_out);

 mux2to1_32b MUX_2(read_data2, sgn_ext_out , alu_src, mux2_out);

 alu ALU(read_data1, mux2_out, alu_ctrl, alu_out, zero);

 shl2 SHL2(sgn_ext_out, shl2_out);

 adder_32b ADDER_2(adder1_out, shl2_out, 1'b0, , adder2_out);

 mux2to1_32b MUX_3(adder1_out, adder2_out, pc_src, mux3_out);

 mux2to1_32b MUX_4(alu_out, data_in, mem_to_reg, mux4_out);

 shl2_26b SHL26(inst[25:0], shl26_out);
 assign in0MUX6 = {adder1_out[31:28], shl26_out};
  mux2to1_32b MUX_6(in0MUX6, read_data1, jr, mux6_out);

 mux2to1_32b MUX_7(mux3_out, mux6_out, jmp, mux7_out);

 assign inst_adr = pc_out;
 assign data_adr = alu_out;  //Address in data_memory where data should be
written
 assign data_out = read_data2; //Value that should be written in data_memory
```

Instruction Formats

Set Less Than Immediate:

```
SLTi Rt Rs data
```

Opcode	R_s	R_t	data
001010	[25:21]	[20:16]	[15:0]

Jump and Link:

```
Jal adr
```

Opcode	adr
000011	[25:0]

Jump Register:

```
Jr Rs
```

Opcode	R_s	X
000110	[25:21]	X

Jump:

```
J adr
```

Opcode	adr
000010	[25:0]

Controller

Issued signals for components in Datapath for each instruction is shown below:

```
  always @(opcode)
   begin
       {reg_dst, alu_src, mem_to_reg, reg_write, mem_read,
mem_write, branch, alu_op, jmp, jr, data_to_write} = 14'd0;
       case (opcode)
           // RType instructions
           6'b000000 : {reg_dst, reg_write, alu_op} =
{2'b01, 3'b110};

           // Load Word (lw) instruction
           6'b100011 : {alu_src, mem_to_reg, reg_write,
mem_read} = 4'b1111;

           // Store Word (sw) instruction
           6'b101011 : {alu_src, mem_write} = 2'b11;

           // Branch on equal (beq) instruction
           6'b000100 : {branch, alu_op} = 3'b101;

           // Add immediate (addi) instruction
           6'b001001: {reg_write, alu_src} = 2'b11;

           // Jump (j) instruction
           6'b000010: {jmp} = 1'b1;

           // Jump and Link (jal) instruction
           6'b000011: {reg_dst, data_to_write, jmp} =
{2'b10, 2'b01, 1'b1};

           // Jump Register (JR) instruction
           6'b000110: {jr, jmp} = {2'b11};

           // Set Less Than immediate (SLTi) instruction
           6'b001010: {reg_write, alu_src,
alu_op,data_to_write} = {1'b1, 1'b1, 2'b11, 2'b10};
       endcase
   end
   assign pc_src = branch & z
```

Testing

The processor is tested using the following test bench:

Load #20 32bit numbers in 2’s complement format from a .mem file into Data Memory and find their minimum and it’s index and finally store them at address 2000 and 2004 of Data Memory.

Pseudocode:

```
min = a[0]
min_idx = 0

for (int i = 1 ; i < 20 ; i++)
      if (a[i] < mn)
      {
            min = a[i]
            min_idx = i
      }
```

The pseudocode above is converted to an equivalent set of Assembly instructions:

Assembly Instructions:

```
           LW R4, 1000(R0)
           ADDi R5, R0, 0
           ADDi R1, R0, 1
           ADDi R2, R0, 20

iLoop:     Beq R1, R2, AFTER_LOOP
           ADD R3, R0, R1
           ADD R3, R3, R3
           ADD R3, R3, R3
```

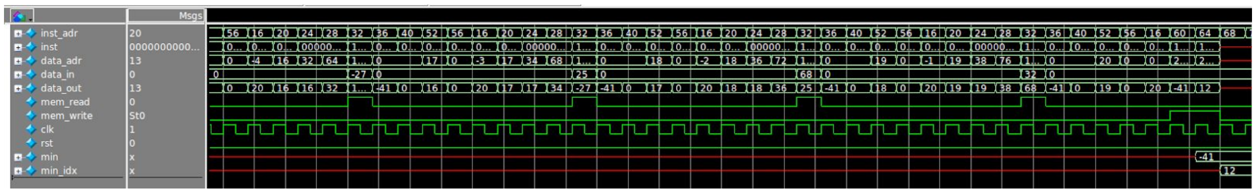
```
            LW R7, 1000(R3)
            SLT R6, R7, R4
            Beq R6, R0, END_LOOP
            ADD R4, R0, R7
            ADD R5, R0, R1
END_LOOP:   ADDi R1, R1, 1
            J iLoop
AFTER_LOOP:SW R4, 2000(R0)
            SW R5, 2004(R0)
```

The loop variable (i) is stored in R1.
Minimum value (min) is stored in R4.
Minimum value's index (min_idx) is stored in R5.

The array of 20 random signed numbers loaded into Data Memory is as followed:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	27	49	94	42	28	83	113	62	109	73	82	-41	102	63	111	-27	25	68	32

The first row corresponds to the indices while the second row corresponds to the values stored in that index.



As it can be seen, the minimum number in the test set is A[12] = -41 which is shown on *min* and *min_idx* in the waveforms after executing all instructions.