III.Verilog Implementation/Design/Verification:

A.Source Code of Top Level:

1.Top Level for MIPS Processor:

```
1  `timescale 1ns / 1ps
2  /****************************** C E C S  4 4 0 ******************************
3   *
4   * File Name:  Lab6_MCU_TopLevel.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 28,  2017
10  *
11  * Purpose:   Top Level containing 5 modules: Instruction Unit, MCU, Intgr DP,
12  *            Data Memory and IO Memory modules connect to from one module to
13  *            another to allow data flow
14  * Notes:      N/A
15  *
16  **************************************************************************/
17 module Lab6_MCU_TopLevel(clk, rst);
18       input    clk, rst;  //system inputs
19
20       //internal wire
21       wire     c_wire, v_wire, n_wire, z_wire, intr_wire, int_ack_wire;
22       wire     [1:0]  Mem_wr_Sel_wire;
23       wire     [31:0] se_16_wire, ALU_OUT_wire, intgr_D_OUT_wire, DY_wire;
24       wire     [31:0] PC_out_wire, IR_out_wire, dMem_D_out_wire, ioMem_D_out_wire;
25       wire     [31:0] pc_mux_out_wire, mem_wr_D_in;
26       wire     [4:0]  s_addr_wire, t_addr_wire, d_addr_wire;
27
28       wire     [4:0]    Flags_to_DP_wire;
29       wire     [4:0]    FS_wire, shift_amount_wire;
30       wire     [2:0]    Y_Sel_wire;
31       wire     [1:0]    D_Sel_wire, pc_sel_wire;
32       wire              pc_ld_wire, pc_inc_wire, ir_ld_wire;
33       wire              im_cs_wire, im_rd_wire, im_wr_wire;
34       wire              D_En_wire, S_Sel_wire, T_Sel_wire, HILO_ld_wire, flg_wb_Sel_wire;
35       wire              dm_cs_wire, dm_rd_wire, dm_wr_wire;
```

```
35       wire              dm_cs_wire, dm_rd_wire, dm_wr_wire;
36       wire              io_cs_wire, io_rd_wire, io_wr_wire;
37
38       //Assignmeng for PC_Mux
39       assign   pc_mux_out_wire = (pc_sel_wire == 0) ?
40                                  {PC_out_wire + {se_16_wire[29:0], 2'b00}}:
41                                  (pc_sel_wire == 1) ?
42                                  {PC_out_wire[31:28], IR_out_wire[25:0], 2'b00}:
43                                  (pc_sel_wire == 2) ? ALU_OUT_wire :  32'h0;
44
45       Intruction_Unit   instrct_unit (
```

```
44
45      Intruction_Unit   instrct_unit (
46                  .clk(clk), .rst(rst),
47                  .pc_ld(pc_ld_wire), .pc_inc(pc_inc_wire),
48                  .im_cs(im_cs_wire), .im_wr(im_wr_wire), .im_rd(im_rd_wire),
49                  .ir_ld(ir_ld_wire),
50                  .PC_in(pc_mux_out_wire), .D_In(), .PC_out(PC_out_wire),
51                  .SE_16(se_16_wire),
52                  .IR_out(IR_out_wire));
53
54      MCU       mcu(
55                  .clk(clk), .rst(rst), .intr(intr_wire),
56                  .c(c_wire), .n(n_wire), .z(z_wire), .v(v_wire),
57                  .IR(IR_out_wire),
58                  .int_ack(int_ack_wire),
59                  .FS(FS_wire),
60                  .SA(shift_amount_wire),
61                  .pc_sel(pc_sel_wire), .pc_ld(pc_ld_wire), .pc_inc(pc_inc_wire),
62                  .ir_ld(ir_ld_wire),
63                  .im_cs(im_cs_wire), .im_rd(im_rd_wire), .im_wr(im_wr_wire),
64                  .D_En(D_En_wire), .D_Sel(D_Sel_wire),
65                  .S_Sel(S_Sel_wire), .T_Sel(T_Sel_wire),
66                  .HILO_ld(HILO_ld_wire), .Y_Sel(Y_Sel_wire),
67                  .Mem_wr_Sel(Mem_wr_Sel_wire), .flg_wb_Sel(flg_wb_Sel_wire),
68                  .dm_cs(dm_cs_wire), .dm_rd(dm_rd_wire), .dm_wr(dm_wr_wire),
69                  .io_cs(io_cs_wire), .io_rd(io_rd_wire), .io_wr(io_wr_wire),
70                  .Flags_to_DP(Flags_to_DP_wire));
71
72      //Assignming for D_Addr
73      assign    d_addr_wire = (D_Sel_wire == 0) ? IR_out_wire[15:11] :
74                              (D_Sel_wire == 1) ? IR_out_wire[20:16] :
75                              (D_Sel_wire == 2) ? 5'd31              :
76                              (D_Sel_wire == 3) ? 5'd29              :
77                                                  IR_out_wire[15:11];
78
79      //Assignment for S_Addr and T_Addr
80      assign    s_addr_wire = IR_out_wire[25:21];
81      assign    t_addr_wire = IR_out_wire[20:16];
82
```

```
80        assign   s_addr_wire = IR_out_wire[25:21];
81        assign   t_addr_wire = IR_out_wire[20:16];
82
83        //Selection input for DY between data from I/O mem or dmMem
84        assign   DY_wire = (io_cs_wire == 1)? ioMem_D_out_wire :dMem_D_out_wire;
85
86        Intgr_Path_2      intgr_p2 (
87                    .clk(clk), .reset(rst),
88                    .DT(se_16_wire),
89                    .DY(DY_wire), .DFlag({27'b0, Flags_to_DP_wire}), //Flgs_to_DP_wire = 5'b
90                    .Mem_wr_Sel(Mem_wr_Sel_wire), .flg_wb_Sel(flg_wb_Sel_wire),
91                    .PC_in(PC_out_wire),
92                    .D_En(D_En_wire),
93                    .D_Addr(d_addr_wire),
94                    .S_Addr(s_addr_wire), .S_Sel(S_Sel_wire),
95                    .T_Addr(t_addr_wire),
96                    .T_Sel(T_Sel_wire),
97                    .FS(FS_wire),
98                    .shift_amount(shift_amount_wire),
99                    .HILO_ld(HILO_ld_wire),
100                   .Y_Sel(Y_Sel_wire),
101                   .C(c_wire), .V(v_wire), .N(n_wire), .Z(z_wire),
102                   .ALU_OUT(ALU_OUT_wire),
103                   .D_OUT(intgr_D_OUT_wire));
104
105       Data_Memory       data_mem (
106                   .clk(clk), .rst(rst),
107                   .dm_cs(dm_cs_wire), .dm_wr(dm_wr_wire), .dm_rd(dm_rd_wire),
108                   .Address(ALU_OUT_wire[11:0]),
109                   .D_in(intgr_D_OUT_wire), .D_Out(dMem_D_out_wire));
110
111       IO_Memory         io_mem(
112                   .clk(clk), .rst(rst),
113                   .intr(intr_wire), .int_ack(int_ack_wire),
114                   .io_cs(io_cs_wire), .io_wr(io_wr_wire), .io_rd(io_rd_wire),
115                   .Address_io(ALU_OUT_wire[11:0]),
116                   .D_in_io(intgr_D_OUT_wire), .D_Out_io(ioMem_D_out_wire));
117   endmodule
118
```

B.Source Code of Verilog modules:

2.Instruction Unit:

```
1   `timescale 1ns / 1ps
2   /****************************** C E C S  4 4 0 ******************************
3    *
4    * File Name:  Instruction_Uniut.v
5    * Project:    CECS 440 Senior Project Customized MIPS Processor
6    * Designer:   Juie Kim
7    * Email:      chealy_ljh@yahoo.co.uk
8    * Rev. No.:   Version 1.0
9    * Rev. Date:  November 28,  2017
10   *
11   * Purpose:    Sub module Intruction Unit contains Program Counter register
12   *             Intruction Memory and IR register to hold the instruction.
13   * Notes:      N/A
14   *
15   **********************************************************************/
16  module Intruction_Unit (clk, rst, pc_ld, pc_inc, im_cs, im_wr, im_rd, ir_ld,
17                          PC_in, D_In, PC_out, SE_16, IR_out);
```

```
16  module Intruction_Unit (clk, rst, pc_ld, pc_inc, im_cs, im_wr, im_rd, ir_ld,
17                          PC_in, D_In, PC_out, SE_16, IR_out);
18
19        input    clk, rst, im_cs, im_wr, im_rd, pc_ld, pc_inc, ir_ld;
20        input         [31:0] PC_in, D_In;
21        output wire   [31:0] SE_16;
22        output wire   [31:0] PC_out, IR_out;
23
24        //internal wires
25        wire     [11:0]   Address_wire;
26        wire     [31:0]   pc_out_wire, D_Out_wire, qOut_ir_wire;
27
28        PC_Program_Counter   pc_cnter(
29                    .clk(clk),
30                    .rst(rst),
31                    .pc_ld(pc_ld),
32                    .pc_inc(pc_inc),
33                    .pc_in(PC_in),
34                    .pc_out(pc_out_wire));
35
36        //assign lower 12 bit of the pc_out as address to instrct mem
37        assign   Address_wire = pc_out_wire[11:0];
38        assign   PC_out       = pc_out_wire;
39
40        Instruction_Memory   instrct_mem(
41                    .clk(clk),
42                    .rst(rst),
43                    .im_cs(im_cs),
44                    .im_wr(im_wr),
45                    .im_rd(im_rd),
46                    .Address(Address_wire),
47                    .D_in(D_In),
48                    .D_Out(D_Out_wire));
49
50        IR_Register    ir_reg(
51                    .clk(clk),
52                    .rst(rst),
53                    .load_ir(ir_ld),
54                    .din_ir(D_Out_wire),
54                      .din_ir(D_Out_wire),
55                      .qOut_ir(qOut_ir_wire));
56
57         assign   IR_out = qOut_ir_wire;
58
59         //Sign_Extend 16 function
60         assign   SE_16  = {{16{qOut_ir_wire[15]}}, qOut_ir_wire[15:0]};
61
62  endmodule
63
```

3.Program Counter:

```
1  `timescale 1ns / 1ps
2  /******************************* C E C S  4 4 0 ********************************
3   *
4   * File Name:  PC_Program_Counter.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 5, 2017
10  *
11  * Purpose:    Sub module loadable register to, 32 bit to interface with the
12  *             Instruction memory module. It has the control to increament
13  *             the output by 4 if needed.
14  * Notes:      N/A
15  *
16  *************************************************************************************/
17 module PC_Program_Counter(clk, rst, pc_ld, pc_inc, pc_in, pc_out);
18
19    input    clk, rst;
20    input    pc_ld, pc_inc;
21    input    [31:0]   pc_in;
22    output   [31:0]   pc_out;
23
24    reg      [31:0]   pc_out;
25
26    always @(posedge clk, posedge rst)
27       if (rst)
28          pc_out <=   32'h0;
29       else
30          case ({pc_ld, pc_inc})
31             2'b00: begin
32                pc_out = pc_out;
33             end
34             2'b01: begin
35                pc_out = pc_out + 4;
36             end
37             2'b10: begin
38                pc_out = pc_in;
39             end
40             2'b11: begin
41                pc_out = pc_out;
42             end
43       endcase  //end case statement
44 endmodule
45
```

4. Instruction Memory

```
/******************************** C E C S  4 4 0 ********************************
 *
 * File Name:  Instruction_Memory.v
 * Project:    CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  November 5, 2017
 *
 * Purpose:    Sub module memory module of a 4096x8 byte addressable memory
 *             big endian format which has chip select, read and write
 *             control.
 * Notes:      N/A
 *
 *******************************************************************************/
module Instruction_Memory(clk, rst, im_cs, im_wr, im_rd, Address, D_in, D_Out);
      input    clk, rst, im_cs, im_wr, im_rd;
      input    [11:0] Address;
      input    [31:0] D_in;
      output   [31:0] D_Out;

      reg      [7:0]  ir_Mem [0:4096]; //1K memory byte addressable

      //writing to ir_Memory is synchronous
      always @(posedge clk, posedge rst)
         if(rst)
            ir_Mem[Address] <= ir_Mem[Address];

         else if (im_cs==1 && im_wr==1) begin
            ir_Mem[Address + 0] <= D_in[31:24];
            ir_Mem[Address + 1] <= D_in[23:16];
            ir_Mem[Address + 2] <= D_in[15:8];
            ir_Mem[Address + 3] <= D_in[7:0];
            end

         else
            ir_Mem[Address[11:0]] <= ir_Mem[Address[11:0]];

```

```
         else
            ir_Mem[Address[11:0]] <= ir_Mem[Address[11:0]];

      //reading from jl_Mem is asynchronous
      assign D_Out = (im_cs==1 && im_rd==1) ?
                     {ir_Mem[Address + 0], ir_Mem[Address + 1],
                      ir_Mem[Address + 2], ir_Mem[Address + 3]} : D_Out;
endmodule

```

5.IR Register:

```
`timescale 1ns / 1ps
/******************************** C E C S  4 4 0 ********************************
 *
 * File Name:  IR_Intrct_Mem_Register.v
 * Project:    CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  November 5, 2017
 *
 * Purpose:    Sub module loadable register, 32 bit to interface with the
 *             Instruction memory module. It has the loadable control to
 *             load intruction 32 bit from the Instruction Memory module
 * Notes:      N/A
 *
 *******************************************************************************/
module IR_Register(clk, rst, load_ir, din_ir, qOut_ir);

      input    clk, rst, load_ir;
      input    [31:0]   din_ir;
      output   [31:0]   qOut_ir;

      reg      [31:0]   qOut_ir;

      always @(posedge clk, posedge rst)
         if (rst)
            qOut_ir <= 32'h0;
         else if (load_ir)
            qOut_ir <= din_ir;
endmodule
```

6.Interger Datapath:

```
`timescale 1ns / 1ps
/****************************** C E C S  4 4 0 ******************************
 *
 * File Name:  Intgr_Path_2.v
 * Project:    CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  Novembee 28,  2017
 *
 * Purpose:    Top level to tight ALU, two loadagle registers, and four
 *             register used as pipelining, and output result throught
 *             the ALU_OUT
 * Notes:      N/A.
 *
 *******************************************************************************/
module Intgr_Path_2(clk, reset, DT, DY, DFlag, Mem_wr_Sel, flg_wb_Sel,
                        PC_in, D_En, D_Addr, S_Addr, S_Sel,
                        T_Addr, T_Sel, FS, shift_amount, HILO_ld, Y_Sel,
                        C, V, N, Z,  ALU_OUT, D_OUT);

```

```
21
22      input                   clk, reset, D_En, T_Sel, S_Sel, HILO_ld, flg_wb_Sel;
23      input           [1:0]   Mem_wr_Sel;
24      input           [2:0]   Y_Sel;
25      input           [4:0]   D_Addr, S_Addr, T_Addr, FS, shift_amount;
26      input           [31:0]  DY, DT, PC_in, DFlag;
27      output wire             C, V, N, Z;
28      output wire     [31:0]  ALU_OUT, D_OUT;
29
30      //declaring internal wires
31      wire            [31:0]  S_output, T_output, Y_hi, Y_lo, HI_out, LO_out;
32      wire                    load_hi, load_lo;
33
34      //internal wires for four pipelining registers
35      wire            C_wire, V_wire, N_wire, Z_wire;
36      wire            [4:0]   S_Addr_wire;
37      wire            [31:0]  RS_output, RT_output, ALU_Out_output, D_in_output;
38      wire            [31:0]  T_MUX_output;
39
40      //selection to load into HI and LO registers
41      assign load_hi  = (HILO_ld)? 1:0;
42      assign load_lo  = (HILO_ld)? 1:0;
43
44      //selection for T_MUX and S_Addr
45      assign T_MUX_output = (T_Sel == 1)? DT    : T_output;
46      assign S_Addr_wire  = (S_Sel == 1)? 5'd29 : S_Addr;
47
48      //Assign Flags input from D_in to each flag output when writing back to
49      //Flags register in the MCU
50      assign {C, V, N, Z} = (flg_wb_Sel == 1) ?
51                            {ALU_OUT[3], ALU_OUT[2], ALU_OUT[1], ALU_OUT[0]} :
52                            {C_wire, V_wire, N_wire, Z_wire};
53
54      RS_register    rs_rg(
55                        .clk(clk),
56                        .rst(reset),
57                        .din(S_output),
58                        .qOut(RS_output));
59
```

```
59
60       RT_register    rt_rg(
61                          .clk(clk),
62                          .rst(reset),
63                          .din(T_MUX_output),
64                          .qOut(RT_output));
65
66       ALU_OUT_reg    alu_out_rg(
67                          .clk(clk),
68                          .rst(reset),
69                          .din(Y_lo),
70                          .qOut(ALU_Out_output));
```

```
70                          .qOut(ALU_Out_output));
71
72      D_in_register  d_in_rg(
73                          .clk(clk),
74                          .rst(reset),
75                          .din(DY),
76                          .qOut(D_in_output));
77
78      regfile32       reg_file(
79                          .clk(clk),
80                          .rst(reset),
81                          .D(ALU_OUT),
82                          .S_Addr(S_Addr_wire),
83                          .T_Addr(T_Addr),
84                          .D_Addr(D_Addr),
85                          .D_En(D_En),
86                          .S(S_output),
87                          .T(T_output));
88
89      ALU_32bit       ALU_32bit(
90                          .Opcode(FS),
91                          .shift_amount(shift_amount),
92                          .S(RS_output),
93                          .T(RT_output),
94                          .y_hi(Y_hi),
95                          .y_lo(Y_lo),
96                          .flg_c(C_wire),
97                          .flg_v(V_wire),
98                          .flg_n(N_wire),
99                          .flg_z(Z_wire));
100
101     HI_Loadable_Register  HI_ld_reg(
102                         .clk(clk),
103                         .rst(reset),
104                         .load_hi(load_hi),
105                         .din_hi(Y_hi),
106                         .qOut_hi(HI_out));
107
```

```
107
108     LO_Loadable_Register  LO_ld_reg(
109                       .clk(clk),
110                       .rst(reset),
111                       .load_lo(load_lo),
112                       .din_lo(Y_lo),
113                       .qOut_lo(LO_out));
114
115     //selection for Y_MUX
116     assign ALU_OUT =  (Y_Sel == 0) ? ALU_Out_output  :
117                       (Y_Sel == 1) ? HI_out          :
118                       (Y_Sel == 2) ? LO_out          :
119                       (Y_Sel == 3) ? D_in_output     :
120                       (Y_Sel == 4) ? PC_in  :
121                       (Y_Sel == 5) ? DFlag : ALU_Out_output;
122
123     assign D_OUT   =  (Mem_wr_Sel == 1) ? DFlag :
124                       (Mem_wr_Sel == 2) ? PC_in :
125                       (Mem_wr_Sel == 0) ? RT_output : RT_output;
126
127 endmodule
```

7. RS_Registger

```
`timescale 1ns / 1ps
/******************************** C E C S  4 4 0 ********************************
 *
 * File Name:  RS_register.v
 * Project:    CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  November 28,  2017
 *
 * Purpose:    Sub module to store data fromt he S_Addr register and as an
 *             input to ALU as to do operation needed
 * Notes:      N/A.
 *
 *******************************************************************************/
module RS_register(clk, rst, din, qOut);

   input    clk, rst;
   input    [31:0]   din;
   output   [31:0]   qOut;

   reg      [31:0]   qOut;

   always @(posedge clk, posedge rst)
      if (rst)
         qOut <=  32'h0;
      else
         qOut <=  din;
endmodule
```

8.RT_Register:

```
`timescale 1ns / 1ps
/******************************** C E C S  4 4 0 ********************************
 *
 * File Name:  RT_register.v
 * Project:   CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  November 28,  2017
 *
 * Purpose:    Sub module to store data fromt he T_Addr register and as an
 *             input to ALU as to do operation needed
 * Notes:      N/A.
 *
 *******************************************************************************/
module RT_register(clk, rst, din, qOut);

   input    clk, rst;
   input    [31:0]   din;
   output   [31:0]   qOut;

   reg      [31:0]   qOut;

   always @(posedge clk, posedge rst)
      if (rst)
         qOut <=  32'h0;
      else
         qOut <=  din;
endmodule
```

9.ALU_Out Register:

```
`timescale 1ns / 1ps
/****************************** C E C S  4 4 0 ******************************
 *
 * File Name:  ALU_OUT_reg.v
 * Project:    CECS 440 Senior Project Customized MIPS Processor
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  Novembr 28,  2017
 *
 * Purpose:    Sub module to store data from the ALU operation output from
 *             y_lo port and to be output to the ALU_OUT
 * Notes:      N/A.
 *
 ***************************************************************************/
module ALU_OUT_reg(clk, rst, din, qOut);

   input    clk, rst;
   input    [31:0]   din;
   output   [31:0]   qOut;

   reg      [31:0]   qOut;

   always @(posedge clk, posedge rst)
      if (rst)
         qOut <=  32'h0;
      else
         qOut <=  din;
endmodule
```

10. D_in Register:

```
`timescale 1ns / 1ps
/****************************** C E C S  4 4 0 ******************************
 *
 * File Name:  D_in.v
 * Project:    Lab_Assignment_6
 * Designer:   Juie Kim
 * Email:      chealy_ljh@yahoo.co.uk
 * Rev. No.:   Version 1.0
 * Rev. Date:  Novemger 5, 2017
 *
 * Purpose:    Sub module to store the low 32-bit input and output D_in
 *             register for the purpose of output to ALU_OUT
 * Notes:      N/A.
 *
 ***************************************************************************/
module D_in_register(clk, rst, din, qOut);

   input    clk, rst;
   input    [31:0]   din;
   output   [31:0]   qOut;

   reg      [31:0]   qOut;

   always @(posedge clk, posedge rst)
      if (rst)
         qOut <=  32'h0;
      else
         qOut <=  din;
endmodule

```

11.Register File:

```
1  `timescale 1ns / 1ps
2  /******************************** C E C S  4 4 0 ********************************
3   *
4   * File Name:  regfile32.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  Novemger 28, 2017
10  *
11  * Purpose:    Designing a dual port 32 x 32 register file. The
12  *             register file is used to read data from and write
13  *             data to
14  * Notes:      N/A
15  *
16  *******************************************************************************/
17 module regfile32(clk, rst, D, S_Addr, T_Addr, D_Addr, D_En, S, T);
18    input              clk, rst, D_En;
19    input       [31:0] D;
20    input       [4:0]  S_Addr, T_Addr, D_Addr;
21    output      [31:0] S, T;
22
23    reg         [31:0] reg_mem [0:31];    //32 register each 32 bit long
24
25    always @(posedge clk, posedge rst)
26       if (rst)
27          reg_mem[0] <= 32'h0;
28       else if (D_En == 1 && D_Addr != 0)
29          reg_mem [D_Addr] <= D;
30       else
31          reg_mem[D_Addr] = reg_mem[D_Addr];
32
33    assign      S  = reg_mem[S_Addr];
34    assign      T  = reg_mem[T_Addr];
35 endmodule
36
```

12.ALU_32 Module:

```
1  `timescale 1ns / 1ps
2  /****************************** C E C S  4 4 0 ******************************
3   *
4   * File Name:  ALU_32bit.v
5   * Project:    CECS 440 Senior Project
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 28,  2017
10  *
11  * Purpose:    Top level of three sub module, designed to output the result
12  *             and the four flags of c, v, n, and z.
13  *
14  * Notes:      N/A
15  *
16  **************************************************************************/
17 module ALU_32bit(Opcode, shift_amount, S, T, y_hi, y_lo, flg_c,
18                  flg_v, flg_n, flg_z);
19
20    input         [4:0]   Opcode, shift_amount;
21    input         [31:0]  S;
22    input         [31:0]  T;
23    output wire   [31:0]  y_hi;
24    output wire   [31:0]  y_lo;
25    output wire           flg_c, flg_v, flg_n, flg_z;
26
27    //declaring internal wires
28    wire   mlt_n, mlt_z, div_n, div_z, mip_c, mip_v, mip_n, mip_z;
29    wire   bar_c, bar_v;
30    wire   [63:32]  remndr,  mlt_hi;
31    wire   [31:0]   quotn, mlt_lo, mip_lo, bar_lo;
32
33    //Instantiate the divide multiply and mip modules
34    BARRELSHIFT_32    barrel_shift_32(
35             .Opcode(Opcode),
36             .Shmt(shift_amount),
37             .SData(S), .TData(T), .y_lo(bar_lo),
38             .c(bar_c), .v(bar_v));
39
```

```
39
40     MPY_32  MPY_32(
41              .Opcode(Opcode),
42              .SData(S), .TData(T),
43              .y_hi(mlt_hi), .y_lo(mlt_lo));
44
45     DIV_32  DIV_32(
46              .Opcode(Opcode),
47              .SData(S), .TData (T),
48              .y_hi(remndr), .y_lo(quotn));
49
50     MIP_32  MIP_32(
51              .Opcode(Opcode),
52              .SData(S), .TData(T), .y_lo(mip_lo),
53              .c (mip_c), .v(mip_v));
```

```
52              .SData(S), .TData(T), .y_lo(mip_lo),
53              .c (mip_c), .v(mip_v));
54
55      //equate the signal out of the multiply and divide module
56      //and set the negative and zero flag the the internal wire
57      //of each module of multiply and divide accordingly
58      assign mlt_n        = (Opcode == 5'h1E)? mlt_hi[63]:1'bx;
59      assign div_n        = (Opcode == 5'h1F)? quotn[31] :1'bx;
60      assign mlt_z        = (Opcode == 5'h1E)? ~(|{mlt_hi,mlt_lo}):1'b1;
61      assign div_z        = (Opcode == 5'h1F)? ~(|{remndr, quotn}):1'b1;
62
63      //equate the output of the multiply and divide to the actual
64      //output of the top level, y_hi and y_lo
65      assign {y_hi, y_lo} = (Opcode == 5'h1E)     ? {mlt_hi, mlt_lo}:
66                            (Opcode == 5'h1F)     ? {remndr, quotn} :
67                           ((Opcode == 5'h0C)||
68                            (Opcode == 5'h0D)||
69                           (Opcode == 5'h0E))     ? {32'b0, bar_lo} : {32'b0, mip_lo};
70
71      //equate the all four flags from the internal wires
72      //to the actual output of the top level, flg_c, flg_v, flg_n, flg_z
73      assign flg_c  =  ((Opcode == 5'h0C) || (Opcode == 5'h0D)
74                                          || (Opcode == 5'h0E)) ? bar_c : mip_c;
75
76      assign flg_v  =  ((Opcode == 5'h0C) || (Opcode == 5'h0D)
77                                          || (Opcode == 5'h0E)) ? bar_v : mip_v;
78
79      assign flg_n  = (Opcode == 5'h1E) ? mlt_n :
80                      (Opcode == 5'h1F) ? div_n :
81                  ((Opcode == 5'h0C)||
82                   (Opcode == 5'h0D)||
83                    (Opcode == 5'h0E)) ? bar_lo[31] : mip_lo[31];
84
85      assign flg_z  = (Opcode == 5'h1E) ? mlt_z :
86                      (Opcode == 5'h1F) ? div_z :
87                  ((Opcode == 5'h0C)||
88                   (Opcode == 5'h0D)||
89                    (Opcode == 5'h0E)) ? ~(|bar_lo) : ~(|mip_lo);
90  endmodule
```

13. Barrel Shift Module:

```
1   `timescale 1ns / 1ps
2   /********************************* C E C S  4 4 0 ********************************
3    *
4    * File Name:  BARRELSHIFT_32.v
5    * Project:    CECS 440 Senior Project Customized MIPS Processor
6    * Designer:   Juie Kim
7    * Email:      chealy_ljh@yahoo.co.uk
8    * Rev. No.:   Version 1.0
9    * Rev. Date:  November 28, 2017
10   *
11   * Purpose:    Sub module designed to calculate the multiplication and
12   *             assign the 64 bit output to proper output data bus
13   *
14   * Notes:      N/A
15   *
16   *********************************************************************************/
17  module BARRELSHIFT_32(Opcode, Shmt, SData, TData, y_lo, c, v);
18
19     input     [4:0]    Opcode, Shmt;
20     input     [31:0]   SData, TData;
21
22     output    [31:0]   y_lo;
23     output             c, v;
24     reg       [31:0]   y_lo;
25     reg                c, v;
```

```
27      always @(*) begin
28         c = 0;
29         v = 0;
30         case({Opcode, Shmt})
31      /*****************************************
32      *BARREL SHIFT LEFT LOGICAL 32
33      *****************************************/
34            {5'h0C, 5'd1}:   {c, y_lo} = {TData[31], {TData[30:0], 1'b0}};          //SLL_1
35            {5'h0C, 5'd2}:   {c, y_lo} = {TData[30], {TData[29:0], 2'b0}};          //SLL_2
36            {5'h0C, 5'd3}:   {c, y_lo} = {TData[29], {TData[28:0], 3'b0}};          //SLL_3
37            {5'h0C, 5'd4}:   {c, y_lo} = {TData[28], {TData[27:0], 4'b0}};          //SLL_4
38            {5'h0C, 5'd5}:   {c, y_lo} = {TData[27], {TData[26:0], 5'b0}};          //SLL_5
39            {5'h0C, 5'd6}:   {c, y_lo} = {TData[26], {TData[25:0], 6'b0}};          //SLL_6
40            {5'h0C, 5'd7}:   {c, y_lo} = {TData[25], {TData[24:0], 7'b0}};          //SLL_7
41            {5'h0C, 5'd8}:   {c, y_lo} = {TData[24], {TData[23:0], 8'b0}};          //SLL_8
42            {5'h0C, 5'd9}:   {c, y_lo} = {TData[23], {TData[22:0], 9'b0}};          //SLL_9
43            {5'h0C, 5'd10}:  {c, y_lo} = {TData[22], {TData[21:0], 10'b0}};         //SLL_10
44            {5'h0C, 5'd11}:  {c, y_lo} = {TData[21], {TData[20:0], 11'b0}};         //SLL_11
45            {5'h0C, 5'd12}:  {c, y_lo} = {TData[20], {TData[19:0], 12'b0}};         //SLL_12
46            {5'h0C, 5'd13}:  {c, y_lo} = {TData[19], {TData[18:0], 13'b0}};         //SLL_13
47            {5'h0C, 5'd14}:  {c, y_lo} = {TData[18], {TData[17:0], 14'b0}};         //SLL_14
48            {5'h0C, 5'd15}:  {c, y_lo} = {TData[17], {TData[16:0], 15'b0}};         //SLL_15
49            {5'h0C, 5'd16}:  {c, y_lo} = {TData[16], {TData[15:0], 16'b0}};         //SLL_16
50            {5'h0C, 5'd17}:  {c, y_lo} = {TData[15], {TData[14:0], 17'b0}};         //SLL_17
51            {5'h0C, 5'd18}:  {c, y_lo} = {TData[14], {TData[13:0], 18'b0}};         //SLL_18
52            {5'h0C, 5'd19}:  {c, y_lo} = {TData[13], {TData[12:0], 19'b0}};         //SLL_19
53            {5'h0C, 5'd20}:  {c, y_lo} = {TData[12], {TData[11:0], 20'b0}};         //SLL_20
54            {5'h0C, 5'd21}:  {c, y_lo} = {TData[11], {TData[10:0], 21'b0}};         //SLL_21
55            {5'h0C, 5'd22}:  {c, y_lo} = {TData[10], {TData[9:0],  22'b0}};         //SLL_22
56            {5'h0C, 5'd23}:  {c, y_lo} = {TData[9],  {TData[8:0],  23'b0}};         //SLL_23
57            {5'h0C, 5'd24}:  {c, y_lo} = {TData[8],  {TData[7:0],  24'b0}};         //SLL_24
58            {5'h0C, 5'd25}:  {c, y_lo} = {TData[7],  {TData[6:0],  25'b0}};         //SLL_25
59            {5'h0C, 5'd26}:  {c, y_lo} = {TData[6],  {TData[5:0],  26'b0}};         //SLL_26
60            {5'h0C, 5'd27}:  {c, y_lo} = {TData[5],  {TData[4:0],  27'b0}};         //SLL_27
61            {5'h0C, 5'd28}:  {c, y_lo} = {TData[4],  {TData[3:0],  28'b0}};         //SLL_28
62            {5'h0C, 5'd29}:  {c, y_lo} = {TData[3],  {TData[2:0],  29'b0}};         //SLL_29
63            {5'h0C, 5'd30}:  {c, y_lo} = {TData[2],  {TData[1:0],  30'b0}};         //SLL_30
64            {5'h0C, 5'd31}:  {c, y_lo} = {TData[1],  {TData[0],    31'b0}};         //SLL_31
65
66
67      /*****************************************
68      *BARREL SHIFT RIGHT LOGICAL 32 bit
69      *****************************************/
70            {5'h0D, 5'd1}:   {c, y_lo} = {TData[0],  {1'b0,  TData[31:1]}};          //SRL_1
71            {5'h0D, 5'd2}:   {c, y_lo} = {TData[1],  {2'b0,  TData[31:2]}};          //SRL_2
72            {5'h0D, 5'd3}:   {c, y_lo} = {TData[2],  {3'b0,  TData[31:3]}};          //SRL_3
73            {5'h0D, 5'd4}:   {c, y_lo} = {TData[3],  {4'b0,  TData[31:4]}};          //SRL_4
74            {5'h0D, 5'd5}:   {c, y_lo} = {TData[4],  {5'b0,  TData[31:5]}};          //SRL_5
75            {5'h0D, 5'd6}:   {c, y_lo} = {TData[5],  {6'b0,  TData[31:6]}};          //SRL_6
76            {5'h0D, 5'd7}:   {c, y_lo} = {TData[6],  {7'b0,  TData[31:7]}};          //SRL_7
77            {5'h0D, 5'd8}:   {c, y_lo} = {TData[7],  {8'b0,  TData[31:8]}};          //SRL_8
78            {5'h0D, 5'd9}:   {c, y_lo} = {TData[8],  {9'b0,  TData[31:9]}};          //SRL_9
79            {5'h0D, 5'd10}:  {c, y_lo} = {TData[9],  {10'b0, TData[31:10]}};         //SRL_10
80            {5'h0D, 5'd11}:  {c, y_lo} = {TData[10], {11'b0, TData[31:11]}};         //SRL_11
81            {5'h0D, 5'd12}:  {c, y_lo} = {TData[11], {12'b0, TData[31:12]}};         //SRL_12
82            {5'h0D, 5'd13}:  {c, y_lo} = {TData[12], {13'b0, TData[31:13]}};         //SRL_13
83            {5'h0D, 5'd14}:  {c, y_lo} = {TData[13], {14'b0, TData[31:14]}};         //SRL_14
84            {5'h0D, 5'd15}:  {c, y_lo} = {TData[14], {15'b0, TData[31:15]}};         //SRL_15
85            {5'h0D, 5'd16}:  {c, y_lo} = {TData[15], {16'b0, TData[31:16]}};         //SRL_16
86            {5'h0D, 5'd17}:  {c, y_lo} = {TData[16], {17'b0, TData[31:17]}};         //SRL_17
87            {5'h0D, 5'd18}:  {c, y_lo} = {TData[17], {18'b0, TData[31:18]}};         //SRL_18
88            {5'h0D, 5'd19}:  {c, y_lo} = {TData[18], {19'b0, TData[31:19]}};         //SRL_19
89            {5'h0D, 5'd20}:  {c, y_lo} = {TData[19], {20'b0, TData[31:20]}};         //SRL_20
90            {5'h0D, 5'd21}:  {c, y_lo} = {TData[20], {21'b0, TData[31:21]}};         //SRL_21
91            {5'h0D, 5'd22}:  {c, y_lo} = {TData[21], {22'b0, TData[31:22]}};         //SRL_22
92            {5'h0D, 5'd23}:  {c, y_lo} = {TData[22], {23'b0, TData[31:23]}};         //SRL_23
93            {5'h0D, 5'd24}:  {c, y_lo} = {TData[23], {24'b0, TData[31:24]}};         //SRL_24
94            {5'h0D, 5'd25}:  {c, y_lo} = {TData[24], {25'b0, TData[31:25]}};         //SRL_25
95            {5'h0D, 5'd26}:  {c, y_lo} = {TData[25], {26'b0, TData[31:26]}};         //SRL_26
96            {5'h0D, 5'd27}:  {c, y_lo} = {TData[26], {27'b0, TData[31:27]}};         //SRL_27
97            {5'h0D, 5'd28}:  {c, y_lo} = {TData[27], {28'b0, TData[31:28]}};         //SRL_28
98            {5'h0D, 5'd29}:  {c, y_lo} = {TData[28], {29'b0, TData[31:29]}};         //SRL_29
99            {5'h0D, 5'd30}:  {c, y_lo} = {TData[29], {30'b0, TData[31:30]}};         //SRL_30
100           {5'h0D, 5'd31}:  {c, y_lo} = {TData[30], {31'b0, TData[31]}};            //SRL_31
101
```

```
101
102     /*****************************************
103     *BARREL SHIFT RIGHT ARITHMATIC 32
104     *****************************************/
105           {5'h0E, 5'd1}:  {c, y_lo} = {TData[0],   { TData[31],     TData[31:1]}};     //SRA_1
106           {5'h0E, 5'd2}:  {c, y_lo} = {TData[1],   {{2{TData[31]}},  TData[31:2]}};     //SRA_2
107           {5'h0E, 5'd3}:  {c, y_lo} = {TData[2],   {{3{TData[31]}},  TData[31:3]}};     //SRA_3
108           {5'h0E, 5'd4}:  {c, y_lo} = {TData[3],   {{4{TData[31]}},  TData[31:4]}};     //SRA_4
109           {5'h0E, 5'd5}:  {c, y_lo} = {TData[4],   {{5{TData[31]}},  TData[31:5]}};     //SRA_5
110           {5'h0E, 5'd6}:  {c, y_lo} = {TData[5],   {{6{TData[31]}},  TData[31:6]}};     //SRA_6
111           {5'h0E, 5'd7}:  {c, y_lo} = {TData[6],   {{7{TData[31]}},  TData[31:7]}};     //SRA_7
112           {5'h0E, 5'd8}:  {c, y_lo} = {TData[7],   {{8{TData[31]}},  TData[31:8]}};     //SRA_8
113           {5'h0E, 5'd9}:  {c, y_lo} = {TData[8],   {{9{TData[31]}},  TData[31:9]}};     //SRA_9
114           {5'h0E, 5'd10}: {c, y_lo} = {TData[9],   {{10{TData[31]}}, TData[31:10]}};    //SRA_10
115           {5'h0E, 5'd11}: {c, y_lo} = {TData[10], {{11{TData[31]}}, TData[31:11]}};    //SRA_11
116           {5'h0E, 5'd12}: {c, y_lo} = {TData[11], {{12{TData[31]}}, TData[31:12]}};    //SRA_12
117           {5'h0E, 5'd13}: {c, y_lo} = {TData[12], {{13{TData[31]}}, TData[31:13]}};    //SRA_13
118           {5'h0E, 5'd14}: {c, y_lo} = {TData[13], {{14{TData[31]}}, TData[31:14]}};    //SRA_14
119           {5'h0E, 5'd15}: {c, y_lo} = {TData[14], {{15{TData[31]}}, TData[31:15]}};    //SRA_15
120           {5'h0E, 5'd16}: {c, y_lo} = {TData[15], {{16{TData[31]}}, TData[31:16]}};    //SRA_16
121           {5'h0E, 5'd17}: {c, y_lo} = {TData[16], {{17{TData[31]}}, TData[31:17]}};    //SRA_17
122           {5'h0E, 5'd18}: {c, y_lo} = {TData[17], {{18{TData[31]}}, TData[31:18]}};    //SRA_18
123           {5'h0E, 5'd19}: {c, y_lo} = {TData[18], {{19{TData[31]}}, TData[31:19]}};    //SRA_19
124           {5'h0E, 5'd20}: {c, y_lo} = {TData[19], {{20{TData[31]}}, TData[31:20]}};    //SRA_20
125           {5'h0E, 5'd21}: {c, y_lo} = {TData[20], {{21{TData[31]}}, TData[31:21]}};    //SRA_21
126           {5'h0E, 5'd22}: {c, y_lo} = {TData[21], {{22{TData[31]}}, TData[31:22]}};    //SRA_22
127           {5'h0E, 5'd23}: {c, y_lo} = {TData[22], {{23{TData[31]}}, TData[31:23]}};    //SRA_23
128           {5'h0E, 5'd24}: {c, y_lo} = {TData[23], {{24{TData[31]}}, TData[31:24]}};    //SRA_24
129           {5'h0E, 5'd25}: {c, y_lo} = {TData[24], {{25{TData[31]}}, TData[31:25]}};    //SRA_25
130           {5'h0E, 5'd26}: {c, y_lo} = {TData[25], {{26{TData[31]}}, TData[31:26]}};    //SRA_26
131           {5'h0E, 5'd27}: {c, y_lo} = {TData[26], {{27{TData[31]}}, TData[31:27]}};    //SRA_27
132           {5'h0E, 5'd28}: {c, y_lo} = {TData[27], {{28{TData[31]}}, TData[31:28]}};    //SRA_28
133           {5'h0E, 5'd29}: {c, y_lo} = {TData[28], {{29{TData[31]}}, TData[31:29]}};    //SRA_29
134           {5'h0E, 5'd30}: {c, y_lo} = {TData[29], {{30{TData[31]}}, TData[31:30]}};    //SRA_30
135           {5'h0E, 5'd31}: {c, y_lo} = {TData[30], {{31{TData[31]}}, TData[31]}};       //SRA_31
136           endcase
137     end
138 endmodule
```

14.MPY_32 Module:

```
1  `timescale 1ns / 1ps
2  /********************************* C E C S  4 4 0 ******************************
3   *
4   * File Name:   MPY_32.v
5   * Project:     CECS 440 Senior Project Customized MIPS Processor
6   * Designer:    Juie Kim
7   * Email:       chealy_ljh@yahoo.co.uk
8   * Rev. No.:    Version 1.0
9   * Rev. Date:   November 28,  2017
10  *
11  * Purpose:     Sub module designed to calculate the multiplication and
12  *              assign the 64 bit output to proper output data bus
13  *
14  * Notes:       N/A
15  *
16  ***************************************************************************/
17 module MPY_32(Opcode, SData, TData, y_hi, y_lo);
18    input         [31:0] SData;
19    input         [31:0] TData;
20    input          [4:0] Opcode;
21    output reg   [63:32] y_hi;
22    output reg    [31:0] y_lo;
23
24    //declare internal wire to temperary hold the 64_bit output
25    wire          [63:0] result_64;
26
27    //declare temperary integer variable
28    integer       tempA = 0;
29    integer       tempB = 0;
30
31    //transfer input data to the integer variable
32    always @(*)
33       begin
34          tempA = SData;
35          tempB = TData;
36       end
37
38    assign result_64 = tempA * tempB;
```

```
38      assign result_64 = tempA * tempB;
39
40      //assign the 64_bit result to proper bit
41      always @ (*)
42      begin
43          case (Opcode)
44          5'h1E: begin
45          y_hi         = result_64[63:32];
46          y_lo         = result_64[31:0];
47          end
48          endcase
49      end
50  endmodule
51
```

15. DIV Module:

```
1  `timescale 1ns / 1ps
2  /****************************** C E C S  4 4 0 ******************************
3   *
4   * File Name:  DIV_32.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 28,  2017
10  *
11  * Purpose:    Sub module to calculate the divsion and assign the
12  *             output to a proper bits
13  *
14  * Notes:      N/A
15  *
16  **************************************************************************/
17 module DIV_32(Opcode, SData, TData, y_hi, y_lo);
18    input       [31:0] SData;
19    input       [31:0] TData;
20    input        [4:0] Opcode;
21    output reg [63:32] y_hi;
22    output reg  [31:0] y_lo;
23
24    //Declare temporay integer variable
25    integer     tempA, tempB;
26
27    //save both input in the integer type variable
28    always @(*)
29       begin
30          tempA = SData;
31          tempB = TData;
32       end
33
34    //operating division and assign to proper bits
35    always @ (*)
36    begin
37       case (Opcode)
38       5'h1F: begin
39       y_hi       = tempA % tempB;
```

```
36      begin
37          case (Opcode)
38          5'h1F: begin
39          y_hi         = tempA % tempB;
40          y_lo         = tempA / tempB;
41          end
42          endcase
43      end
44  endmodule
45
```

16. MIP_32 Module:

```
1  `timescale 1ns / 1ps
2  /****************************** C E C S  4 4 0 ******************************
3   *
4   * File Name:  MIP_32.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 28, 2017
10  *
11  * Purpose:    Sub module designed to operate unary operand and two
12  *             operand. This a standard mip architecture operations
13  *
14  * Notes:      N/A
15  *
16  ***************************************************************************/
17 module MIP_32(Opcode, SData, TData, y_lo, c, v);
18    input       [31:0] SData;
19    input       [31:0] TData;
20    input        [4:0] Opcode;
21    output reg  [31:0] y_lo;
22    output reg  c, v;
23
24    //internal wire to hold 33 bit of the result of the addition and
25    //substruction
26    reg          [32:0] tempData;
27    reg          [31:0] sd, td;
28
29    //Declare temporay integer variable
30    integer     tempA, tempB;
31
32    //parameter declaration
33    parameter    [4:0] Pass_S  = 5'h00,
34                       Pass_T  = 5'h01,
35                       ADD     = 5'h02,
36                       ADDU    = 5'h03,
37                       SUB     = 5'h04,
38                       SUBU    = 5'h05,
39                       SLT     = 5'h06,
```

```
38                       SUBU    = 5'h05,
39                       SLT     = 5'h06,
40                       SLTU    = 5'h07,
41                       AND     = 5'h08,
42                       OR      = 5'h09,
43                       XOR     = 5'h0A,
44                       NOR     = 5'h0B,
45                       SLL     = 5'h0C,
46                       SRL     = 5'h0D,
47                       SRA     = 5'h0E,
48                       INC     = 5'h0F,
49                       DEC     = 5'h10,
50                       INC4    = 5'h11,
51                       DEC4    = 5'h12,
52                       ZEROS   = 5'h13,
53                       ONES    = 5'h14,
54                       SP_INIT = 5'h15,
55                       ANDI    = 5'h16,
56                       ORI     = 5'h17,
57                       XORI    = 5'h18,
58                       LUI     = 5'h19,
59                       MUL     = 5'h1E,
60                       DIV     = 5'h1F;
```

```
61
62    //save both input in the integer type variable
63    always @(*)
64       begin
65          tempA = SData;
66          tempB = TData;
67       end
68    //the assignment of each operation according to
69    //the opcode input
70    always @(*)
71    begin
72       c = 0;
73       v = 0;
74       tempData = 33'h0000;
75       case (Opcode)
76          Pass_S:
77             begin
78                y_lo = SData;
79             end
80          Pass_T:
81             begin
82                y_lo = TData;
83             end
84          ADD:
85             begin
86                tempData = {1'b0, SData} + {1'b0, TData};
87                y_lo     = tempData[31:0];
88                c        = tempData[32];
89                if (c == 0)
90                   v = 0;
91                else
92                   v = ~tempData[32];
93             end
94          SUB:
95             begin
96                tempData = {1'b0, SData} - {1'b0, TData};
97                y_lo     = tempData[31:0];
98                c        = tempData[32];
99                if (c == 0)
```

```
99                if (c == 0)
100                  v = 0;
101               else
102                  v = ~tempData[32];
103            end
104         ADDU:
105            begin
106               tempData = {1'b0, SData} + {1'b0, TData};
107               y_lo     = tempData[31:0];
108               c        = tempData[32];
109               v        = tempData[32];
110            end
111         SUBU:
112            begin
113               tempData = {1'b0, SData} - {1'b0, TData};
114               y_lo     = tempData[31:0];
115               c        = tempData[32];
116               v        = tempData[32];
117            end
```

```
118         SLT:
119            begin
120               if ((tempA < tempB)|| (tempA == tempB))
121                  y_lo = 1;
122               else
123                  y_lo = 0;
124            end
125         SLTU:
126            begin
127               if (SData < TData)
128                  y_lo = 1;
129               else
130                  y_lo = 0;
131            end
132         AND:
133            begin
134               y_lo = SData & TData;
135            end
136         OR:
137            begin
138               y_lo = SData | TData;
139            end
140         XOR:
141            begin
142               y_lo = SData ^ TData;
143            end
144         NOR:
145            begin
146               y_lo = ~(SData | TData);
147            end
148         INC:
149            begin
150               tempData = {1'b0, SData} + 1;
151               y_lo     = tempData[31:0];
152            end
153           DEC:
154              begin
155                 tempData = {1'b0, SData} - 1;
156                 y_lo     = tempData[31:0];
157              end
158           INC4:
159              begin
160                 tempData = {1'b0, SData} + 4;
161                 y_lo     = tempData[31:0];
162              end
163           DEC4:
164              begin
165                 tempData = {1'b0, SData} - 4;
166                 y_lo     = tempData[31:0];
167              end
168           ZEROS:
169              begin
170                 y_lo = 32'h0;
171              end
172           ONES:
173              begin
174                 y_lo = 32'hFFFFFFFF;
175              end
```

```
176         SP_INIT:
177            begin
178               y_lo = 32'h3FC;
179            end
180         ANDI:
181            begin
182               y_lo = (SData & {16'h0, TData[15:0]});
183            end
184         ORI:
185            begin
186               y_lo = (SData | {16'h0, TData[15:0]});
187            end
188         XORI:
189            begin
190               y_lo = (SData ^ {16'h0, TData[15:0]});
191            end
192         LUI:
193            begin
194               y_lo = {TData[15:0], 16'h0};
195            end
196       endcase
197    end
198 endmodule
```

17.HILO ladable register:

```
1  `timescale 1ns / 1ps
2  /********************************* C E C S  4 4 0 ********************************
3   *
4   * File Name:  HI_Loadable_Register.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 5, 2017
10  *
11  * Purpose:    Sub module to load the MSB 32-bit output from 32-bit ALU
12  *             operations, store in the register to output MSB 32bit.
13  * Notes:      N/A
14  *
15  *******************************************************************************/
16 module HI_Loadable_Register(clk, rst, load_hi, din_hi, qOut_hi);
17
18    input    clk, rst, load_hi;
19    input    [31:0]   din_hi;
20    output   [31:0]   qOut_hi;
21
22    reg      [31:0]   qOut_hi;
23
24    always @(posedge clk, posedge rst)
25       if (rst)
26          qOut_hi <=  32'h0;
27       else if (load_hi)
28          qOut_hi <=  din_hi;
29 endmodule
```

```
1  `timescale 1ns / 1ps
2  /********************************* C E C S  4 4 0 ******************************
3   *
4   * File Name:  LO_Loadable_Register.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 5, 2017
10  *
11  * Purpose:    Sub module to load the low 32-bit output from 32-bit ALU
12  *             operations, store in the register to output LSB 32bit.
13  * Notes:      N/A.
14  *
15  ***************************************************************************/
16 module LO_Loadable_Register(clk, rst, load_lo, din_lo, qOut_lo);
17
18    input    clk, rst, load_lo;
19    input    [31:0]   din_lo;
20    output   [31:0]   qOut_lo;
21
22    reg      [31:0]   qOut_lo;
23
24    always @(posedge clk, posedge rst)
25       if (rst)
26          qOut_lo <=  32'h0;
27       else if (load_lo)
28          qOut_lo <=  din_lo;
29 endmodule
```

18. Data Memory Module:

```
1  `timescale 1ns / 1ps
2  /********************************* C E C S  4 4 0 ******************************
3   *
4   * File Name:  Data_Memory.v
5   * Project:    CECS 440 Senior Project Customized MIPS Processor
6   * Designer:   Juie Kim
7   * Email:      chealy_ljh@yahoo.co.uk
8   * Rev. No.:   Version 1.0
9   * Rev. Date:  November 28, 2017
10  *
11  * Purpose:    Memory of 1K x 32-bit width. the total address is 1024 or 1K
12  *             the width is 32-bit, which is asynchronous read and synchronous
13  *             write with chip select to perform read and write
14  * Notes:      N/A
15  *
16  ***************************************************************************/
17 module Data_Memory(clk, rst, dm_cs, dm_wr, dm_rd, Address, D_in, D_Out);
18
19       input              clk, rst, dm_cs, dm_wr, dm_rd;
20       input       [11:0] Address;
21       input       [31:0] D_in;
22       output      [31:0] D_Out;
23       wire        [31:0] D_Out;
24
25       reg         [7:0] jl_Mem [0:4095]; //1K memory byte addressable
26
27       //writing to jl_Memory is synchronous
28       always @(posedge clk, posedge rst)
29          if (rst)
30             jl_Mem[Address] <= 8'b0;
```

```
27       //writing to jl_Memory is synchronous
28       always @(posedge clk, posedge rst)
29          if (rst)
30             jl_Mem[Address] <= 8'b0;
31
32          else if (dm_cs==1 && dm_wr==1) begin
33             jl_Mem[Address + 0] <= D_in[31:24];
34             jl_Mem[Address + 1] <= D_in[23:16];
35             jl_Mem[Address + 2] <= D_in[15:8];
36             jl_Mem[Address + 3] <= D_in[7:0];
37             end
38
39          else
40             jl_Mem[Address[11:0]] <= jl_Mem[Address[11:0]];
41
42       //reading from jl_Mem is asynchronous
43       assign D_Out = (dm_cs==1 && dm_rd==1) ?
44                      {jl_Mem[Address + 0], jl_Mem[Address + 1],
45                       jl_Mem[Address + 2], jl_Mem[Address + 3]} : D_Out;
46  endmodule
47
```

19.I/O Memory Module:

```
1   `timescale 1ns / 1ps
2   /****************************** C E C S  4 4 0 ******************************
3    *
4    * File Name:  Data_Memory.v
5    * Project:    CECS 440 Senior Project Customized MIPS Processor
6    * Designer:   Juie Kim
7    * Email:      chealy_ljh@yahoo.co.uk
8    * Rev. No.:   Version 1.0
9    * Rev. Date:  November 28,  2017
10   *
11   * Purpose:    Memory of 1K x 32-bit width. the total address is 1024 or 1K
12   *             the width is 32-bit, which is asynchronous read and synchronous
13   *             write with chip select to perform read from IO module as INPUT and
14   *             write to IO module as OUPUT
15   * Notes:      N/A
16   *
17   ***************************************************************************/
18  module IO_Memory(clk, rst, intr, int_ack, io_cs, io_wr, io_rd, Address_io, D_in_io, D_Out_io);
19        input     int_ack;
20        output reg intr;
21
22        input              clk, rst, io_cs, io_wr, io_rd;
23        input       [11:0] Address_io;
24        input       [31:0] D_in_io;
25        output      [31:0] D_Out_io;
26        wire        [31:0] D_Out_io;
27
28        reg         [7:0] ioMem [0:4095]; //1K memory byte addressable
29
30        initial begin
31           intr = 0;
32  //       #1060
33           #700
34           intr = 1;
35           @(posedge int_ack)
36              intr = 0;
37        end
```

```
38
39       //writing to jl_Memory is synchronous
40       always @(posedge clk, posedge rst)
41          if (rst)
42             ioMem[Address_io] <= 8'b0;
43
44          else if (io_cs==1 && io_wr==1) begin
45             ioMem[Address_io + 0] <= D_in_io[31:24];
46             ioMem[Address_io + 1] <= D_in_io[23:16];
47             ioMem[Address_io + 2] <= D_in_io[15:8];
48             ioMem[Address_io + 3] <= D_in_io[7:0];
49             end
50          else
51             ioMem[Address_io[11:0]] <= ioMem[Address_io[11:0]];
52
53       //reading from io_Mem is asynchronous
54       assign D_Out_io = (io_cs==1 && io_rd==1) ?
55                         {ioMem[Address_io + 0], ioMem[Address_io + 1],
56                          ioMem[Address_io + 2], ioMem[Address_io + 3]} : D_Out_io;
57 endmodule
```

C.Memory Modules used for verification: Data memory and instruction memory used to test the processor

1.Data Memory Moduels:

dMem01_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //0x04
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //0x08
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //0x0C
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem02_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //0x04
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //0x08
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //0x0C
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem03_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //0x04
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //0x08
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //0x0C
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem04_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //0x04
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //0x08
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //0x0C
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem05_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //0x04
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //0x08
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //0x0C
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem06_Fa17.dat - Notepad

File Edit Format View Help

```
@00             // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //word 4
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //word 8
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //word 12
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem07_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

C3 C3 C3 C3     // 0x00:03
12 34 56 78     // 0x04:07
89 AB CD EF     // 0x08:0B
A5 A5 A5 A5     // 0x0C:0F
5A 5A 5A 5A     // 0x10:13  //word 4
24 68 AC E0     // 0x14:17
13 57 9B DF     // 0x18:1B
0F 0F 0F 0F     // 0x1C:1F
F0 F0 F0 F0     // 0x20:23  //word 8
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33  //word 12
00 00 00 0D     // 0x34:37
FF FF FF F8     // 0x38:3B
00 00 75 CC     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem08_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endia

00 00 00 19     // 0x00:03
00 00 03 E8     // 0x04:07
FF FF FF E7     // 0x08:0B
FF FF FC 18     // 0x0C:0F
00 00 61 A8     // 0x10:13
FF FF 9E 58     // 0x14:17
FF FF FF FF     // 0x18:1B
00 00 00 07     // 0x1C:1F
00 00 00 08     // 0x20:23
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33
00 00 00 0D     // 0x34:37
00 00 00 0E     // 0x38:3B
00 00 00 0F     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem09_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

00 04 09 11     // 0x00:03  //word 00 =
00 00 03 E8     // 0x04:07  //word 01 =
FF FB F6 EF     // 0x08:0B  //word 02 = -
FF FF FC 18     // 0x0C:0F  //word 03 =
00 00 01 08     // 0x10:13  //word 04 =
00 00 01 D1     // 0x14:17  //word 05 =
FF FF FE F8     // 0x18:1B  //word 06 =
FF FF FE 2F     // 0x1C:1F  //word 07 =
00 00 00 08     // 0x20:23  //word 08 =
00 00 00 09     // 0x24:27  //word 09 =
00 00 00 0A     // 0x28:2B  //word 10 =
00 00 00 0B     // 0x2C:2F  //word 11 =
00 00 00 0C     // 0x30:33  //word 12 =
00 00 00 0D     // 0x34:37  //word 13 =
00 00 00 0E     // 0x38:3B  //word 14 =
00 00 00 0F     // 0x3C:3F  //word 15 =

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem10_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Format

00 04 09 11     // 0x00:03  //word 00 =  264465
00 00 03 E8     // 0x04:07  //word 01 =    1000
FF FB F6 EF     // 0x08:0B  //word 02 = -264465
FF FF FC 18     // 0x0C:0F  //word 03 =   -1000
00 00 01 08     // 0x10:13  //word 04 =     264
00 00 01 D1     // 0x14:17  //word 05 =     465
FF FF FE F8     // 0x18:1B  //word 06 =    -264
FF FF FE 2F     // 0x1C:1F  //word 07 =
00 00 00 08     // 0x20:23  //word 08 =
00 00 00 09     // 0x24:27  //word 09 =
00 00 00 0A     // 0x28:2B  //word 10 =
00 00 00 0B     // 0x2C:2F  //word 11 =
00 00 00 0C     // 0x30:33  //word 12 =
00 00 00 0D     // 0x34:37  //word 13 =
00 00 00 0E     // 0x38:3B  //word 14 =
00 00 00 0F     // 0x3C:3F  //word 15 =

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem11_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian F

00 04 09 11     // 0x00:03
00 00 03 E8     // 0x04:07
FF FB F6 EF     // 0x08:0B
FF FF FC 18     // 0x0C:0F
00 00 01 08     // 0x10:13
00 00 01 D1     // 0x14:17
FF FF FE F8     // 0x18:1B
FF FF FE 2F     // 0x1C:1F
00 00 00 08     // 0x20:23
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33
00 00 00 0D     // 0x34:37
00 00 00 0E     // 0x38:3B
00 00 00 0F     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem12_Fa17.dat - Notepad

File Edit Format View Help

```
@0              // Big Endian Fo

00 04 09 11     // 0x00:03
00 00 03 E8     // 0x04:07
FF FB F6 EF     // 0x08:0B
FF FF FC 18     // 0x0C:0F
00 00 01 08     // 0x10:13
00 00 01 D1     // 0x14:17
FF FF FE F8     // 0x18:1B
FF FF FE 2F     // 0x1C:1F
00 00 00 08     // 0x20:23
00 00 00 09     // 0x24:27
00 00 00 0A     // 0x28:2B
00 00 00 0B     // 0x2C:2F
00 00 00 0C     // 0x30:33
00 00 00 0D     // 0x34:37
00 00 00 0E     // 0x38:3B
00 00 00 0F     // 0x3C:3F

@1CC
AB CD EF 01     // 0x1CC:1CF

@3F8
00 00 00 00     // 0x3F8:3FB
```

dMem13_Fa17.dat - Notepad

File Edit Format View Help

```
@0

C3 C3 C3 C3
12 34 56 78
89 AB CD EF
A5 A5 A5 A5
5A 5A 5A 5A
24 68 AC E0
13 57 9B DF
0F 0F 0F 0F
F0 F0 F0 F0
00 00 00 09
00 00 00 0A
00 00 00 0B
00 00 00 0C
00 00 00 0D
FF FF FF F8
00 00 75 CC

@1CC
AB CD EF 01

@3F8
00 00 00 00
```

dMem14_Fa17.dat - Notepad

File Edit Format View Help

```
@0

C3 C3 C3 C3
12 34 56 78
89 AB CD EF
A5 A5 A5 A5
5A 5A 5A 5A
24 68 AC E0
13 57 9B DF
0F 0F 0F 0F
F0 F0 F0 F0
00 00 00 09
00 00 00 0A
00 00 00 0B
00 00 00 0C
00 00 00 0D
FF FF FF F8
00 00 75 CC

@1CC
AB CD EF 01

@3F8
00 00 00 00

@3FC
00 00 02 00
```

2. Instruction Memory:

iMem01_Fa17.dat

File Edit Format

```
@0
3c 01 12 34
34 21 56 78
3c 02 87 65
34 42 43 21
00 01 18 20
10 22 00 01
10 23 00 03
3c 0e ff ff
35 ce ff ff
00 00 00 0d
00 00 70 20
14 23 00 01
14 22 00 03
3c 0f ff ff
35 ef ff ff
00 00 00 0d
00 00 78 20
3c 0d 10 01
35 ad 00 c0
ad a1 00 00
00 00 00 0d
```

iMem02_Fa17.dat - N

File Edit Format Vie

```
@0
3c 01 ff ff
34 21 ff ff
20 02 00 10
3c 0f 10 01
35 ef 00 c0
00 01 08 42
ad e1 00 00
21 ef 00 04
20 42 ff ff
14 40 ff fb
08 10 00 0c
00 00 00 0d
3c 0e 5a 5a
35 ce 3c 3c
00 00 00 0d
```

iMem03_Fa17.dat -

File Edit Format V

```
@0
3c 01 80 00
34 21 ff ff
20 02 00 10
3c 0f 10 01
35 ef 00 c0
00 01 08 43
ad e1 00 00
21 ef 00 04
20 42 ff ff
14 40 ff fb
08 10 00 0c
00 00 00 0d
3c 0e 5a 5a
35 ce 3c 3c
00 00 00 0d
```

iMem04_Fa17.dat - Notepad

File Edit Format View Help

```
@0
3c 01 ff ff
34 21 ff ff
20 02 00 10
3c 0f 10 01
35 ef 00 c0
00 01 08 40
ad e1 00 00
21 ef 00 04
20 42 ff ff
00 02 18 2a
14 60 ff fa
08 10 00 0d
00 00 00 0d
3c 0e 5a 5a
35 ce 3c 3c
00 00 00 0d
```