```
1   `timescale 1ns / 1ps
2   /**********************************************************************************
3    *
4    * Author:   Jesus Luciano
5    * Filename: alu_32.v
6    * Date:     1/22/2019
7    * Version:  1.0
8    *
9    * Notes:1.0 Arithmetic Logic Unit with two 32-bit inputs and a 5-bit function
10   *           select input which selects between 28 operations to output to its
11   *           2 32-bit outputs as well as 4 status flags, (C) Carry, (V) Overflow
12   *           (N) Negative, and (Z) Zero.
13   *       1.1 Added shift amount input, shamt, as well as instantiating shift
14   *           module to handle shifts
15   *
16   **********************************************************************************/
17  module alu_32(S, T, FS, C, V, N, Z, Y_hi, Y_lo, shamt);
18
19      //delcare inputs
20      input  [31:0]  S, T;
21      input  [ 4:0] FS, shamt;
22
23      //delcare outputs
24      output [31:0] Y_hi, Y_lo;
25      output        N, Z, V, C;
26
27      //wires for outputs of MIPS_32, MPY_32 and DIV_32 ALUs
28      wire [31:0] main_Y_hi, main_Y_lo,
29                   div_Y_hi,  div_Y_lo,
30                   mpy_Y_hi,  mpy_Y_lo,
31                  shft_Y_hi, shft_Y_lo;
32      //flag outputs of MIPS_32
33      wire        main_V,    main_C,
34                  shft_V,    shft_C;
35
36      //main alu instantiation with explicit port mapping
37      MIPS_32 main_alu(.S(S), .T(T), .FS(FS), .V(main_V), .C(main_C),
38                       .Y_hi(main_Y_hi),      .Y_lo(main_Y_lo) );
39
40      //division module instantiated with explicit port mapping
41      //module yields 32 bit quotient and 32 bit remainder
42      DIV_32   div_alu(.S(S), .T(T), .Y_hi(div_Y_hi), .Y_lo(div_Y_lo) );
43
44      //multiplication module instantiated with explicit port mapping
45      //module yields 64 bit output
46      MPY_32   mpy_alu(.S(S), .T(T), .Y_hi(mpy_Y_hi), .Y_lo(mpy_Y_lo) );
47
48      SHIFT  shift_alu(.T(T), .FS(FS), .shamt(shamt), .V(shft_V), .C(shft_C),
49                       .Y_hi(shft_Y_hi), .Y_lo(shft_Y_lo) );
50
51      //4to1 mux for Y_hi and Y_lo outputs
52      assign {Y_hi,Y_lo} = (FS == 5'h1E) ? { mpy_Y_hi,  mpy_Y_lo} :
53                           (FS == 5'h1F) ? { div_Y_hi,  div_Y_lo} :
54                           (FS == 5'h0C  |
55                            FS == 5'h0D  |
56                            FS == 5'h0E) ? {shft_Y_hi, shft_Y_lo} :
57                                           {main_Y_hi, main_Y_lo} ;
```

```
58
59       //all status flags controlled by 3to1 mux
60                            //multiplication
61       assign {C,V,N,Z} = (FS == 5'h1E) ? {2'bx, Y_hi[31], ({Y_hi, Y_lo} == 64'h0)} :
62                            //division, flags determined by quotient only
63                           (FS == 5'h1F) ? {2'bx, Y_lo[31], (      {Y_lo} == 32'h0)} :
64                           //negative flag always zero for unsigned cases
65                           //addu
66                           (FS == 5'h03) ? {main_C, main_V,    1'b0,(Y_lo == 32'h0)} :
67                           //subu
68                           (FS == 5'h05) ? {main_C, main_V,    1'b0,(Y_lo == 32'h0)} :
69                           //shift cases
70                           (FS == 5'h0C  |
71                            FS == 5'h0D  |
72                            FS == 5'h0E) ? {shft_C, shft_V, Y_lo[31],(Y_lo == 32'h0)}:
73                            //all other cases
74                                          {main_C, main_V, Y_lo[31],(Y_lo == 32'h0)};
75
76
77   endmodule
```