1. Arithmetic Logic Units

5.9 Design the 32-bit ALU shown in figure 5.15 using HDL
5.10 Add an overflow output to the 32-bit ALU
5.11 Add a zero output to the 32-bit ALU

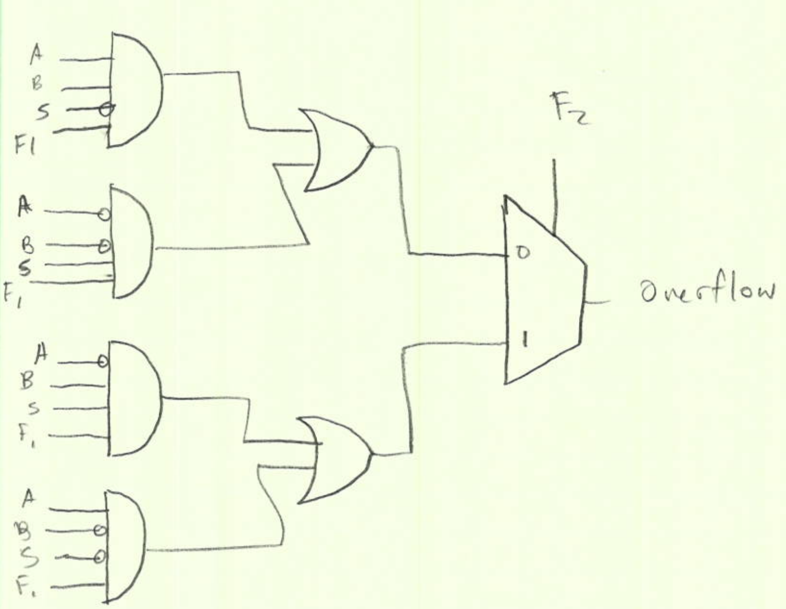
```
module  32bit_ALU (input [31:0] A,B,
                   input [2:0] F,
        output reg [31:0] y,
        output Cout,
        output Zero, Overflow);  // added in for 5.10
        wire [32:0] S;
        wire [31:0] Bout;

        assign Bout = F[2] ? ~B : B;
        assign S= A+ Bout + F[2];
        assign Cout = S[32];

        always @ (*)
            case (F[1:0])
                2'b00: Y <= A & Bout;
                2'b01: Y <= A | Bout;
                2'b10: Y <= S;
                2'b11: Y <= S[32];
            endcase

        assign Zero = (Y == 32'b0);

        always @ (*)
            case (F[2:1])
              2'b01: overflow <= A[31] & B[31] & ~S[31] |
                                  ~A[31] & ~B[31] & S[31];
              2'b11: overflow <= ~A[31] & B[31] & S[31] |
                                   A[31] & ~B[31] & ~S[31];
              default: Overflow <= 1'b0;
            endcase
endmodule
```

5.10 a) write a boolean equation for the Overflow output

when f[2:1] = 01 : $AB\overline{S} + \overline{A}\overline{B}S$ = overflow

when f[2:1] = 11 : $\overline{A}BS + A\overline{B}\overline{S}$ = overflow

b) Sketch the overflow circuit



5.29 Express the base 10 numbers in 5.25 in IEEE 754 floating point.

a. -13.5625 → -1101.1001 = -1.1011001 × 2^3

$1000\,0010_2$

1 100|0 001|0 101|1001 |0000|0000|0000|0000

C 1 5 9 0 0 0 0

= $\boxed{C1590000}$

b. 42.3125 → 101010.0101 = 1.010100101 × 2^5

$1\,000\,0100_2$

0 1 00|0010|0 010|1001|01 00 |0000|0000|0000

4 2 2 9 4 0 0 0

= $\boxed{42294000}$

c. -17.15625 → -10001.00101 = -1.000100101 × 2^4

$1000\,0011_2$

1 100|000 1|1000|1001|0100/0000|0000|0000

C 1 8 9 4 0 0 0

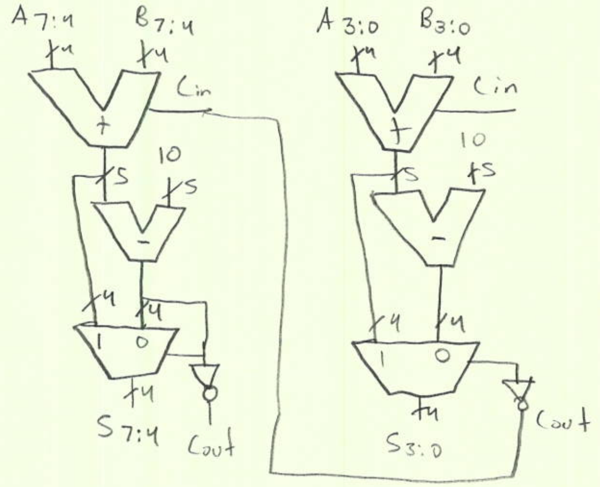
= $\boxed{C1894000}$

Interview Questions

5.2 Binary coded decimal representation uses four bits to encode each decimal digit. Explain why processors might use BCD representation.

Binary coded decimal can be useful in representation of more accurate decimal numbers and ease of use when converting into human-readable representations. Decimal fixed points and floating points are important in comercial and industrial computing.

5.3 Design hardware to add two8-bit unsigned BCD numbers



```
module 8bit_BCD (input [7:0] a,b,
                 input cin,
                 output [7:0] s,
                 output cout);

    wire carry c0;

    4_bitadder0(a [3:0], b[3:0], cin , s[3:0] , c0);
    4_bitadder1 (a [7:4], c0 , s[7:4], cout);
endmodule

module 4 bit_BCD ( imput [3:0] a,b
                   input cin,
                   output [3:0] s,
                   output cout);

        wire [4:0] result, sub10;

        assign result = a+b+cin;
        assign sub10 = result - 10;

        assign cout = ~ sub10[4];
        assign s = sub10[4] ? result [3:0] : sub10 [3:0];
endmodule
```