Main module:

```
module main(input clk, input B1, B2, B3, B4, B5, B6, B7, B8, B9,  output [7:0]seg, output [3:0] anoact);
wire outClk;
clkDivider #(250000) gh(1'b0,clk,outClk);
wire [1:0] count;
counterN #(2,4) ccs(outClk,1'b0,1'b1,count);

reg [3:0]num_1_tens;
reg [3:0]num_1_units;
reg [3:0]num_2_tens;
reg [3:0]num_2_units;
reg [3:0]current_num;
reg [13:0] result;
reg target;
reg is_neg_thousands;
reg is_neg_tens;
reg div_error;
reg [3:0] result_units;
reg [3:0] result_tens;
reg [3:0] result_hundreds;
reg [3:0] result_thousands;

sevenSeg SS(1'b1,current_num,count,seg,anoact,target);

always@(*)begin
   case(count)
      0: current_num = result_units;
      1: current_num = result_tens;
      2: current_num = result_hundreds;
```

```
      3: current_num = result_thousands;
   endcase
end

always@(posedge B1)
begin
num_1_tens = num_1_tens + 1;
if(num_1_tens == 10) num_1_tens = 0;
end

always@(posedge B2)
begin
num_1_units = num_1_units +1;
if(num_1_units == 10) num_1_units =0;
end

always@(posedge B3)
begin
num_2_tens = num_2_tens +1;
if(num_2_tens == 10) num_2_tens =0;
end

always@(posedge B4)
begin
num_2_units = num_2_units +1;
if(num_2_units == 10) num_2_units =0;
end

always@(posedge B5 or posedge B6 or posedge B7 or posedge B8)
```

```
begin
   if(B5)
      begin
       result = num_1_tens*10 + num_1_units + num_2_tens * 10 + num_2_units;
       is_neg_thousands = 0;
       is_neg_tens = 0;
       div_error = 0;
      end
   else if(B6)
      begin
       div_error = 0;
            if(num_1_tens>num_2_tens)
          begin
                        result = (num_1_tens*10 + num_1_units) - (num_2_tens * 10 + num_2_units);
                        is_neg_thousands = 0;
                        is_neg_tens = 0;
          end
           else if(num_1_tens<num_2_tens)
          begin
                        result = (num_2_tens*10 + num_2_units) - (num_1_tens * 10 + num_1_units);
                        if(result > 9)
                        begin
                                is_neg_thousands = 1;
                                is_neg_tens = 0;
                        end
                        else if(result  < 10)
                        begin
                                is_neg_tens = 1;
                                is_neg_thousands = 0;
```

```
                    end
        end
         else if(num_1_tens==num_2_tens)
                begin
                 if(num_1_units>num_2_units)
                  begin
                    result = (num_1_tens*10 + num_1_units) - (num_2_tens * 10 + num_2_units);
                        is_neg_thousands = 0;
                        is_neg_tens = 0;
                  end
                 else if(num_1_units<num_2_units)
                  begin
                    result = (num_2_tens*10 + num_2_units) - (num_1_tens * 10 + num_1_units);
          is_neg_tens = 1;
          is_neg_thousands = 0;
                  end
                 else if(num_1_units==num_2_units)
                  begin
                    result = (num_1_tens*10 + num_1_units) - (num_2_tens * 10 + num_2_units);
                    is_neg_thousands = 0;
                    is_neg_tens = 0;
                  end
                end
            end

else if(B7)
  begin
    result = ((num_1_tens * 10 + num_1_units) * (num_2_tens * 10 + num_2_units));
    is_neg_thousands = 0;
```

```
      is_neg_tens = 0;
      div_error = 0;
    end
  else if(B8)
    begin
      if((num_2_tens * 10 + num_2_units) != 0)
        begin
          if(((num_1_tens*10 + num_1_units) % (num_2_tens * 10 + num_2_units)) >= ((num_2_tens *
10 + num_2_units)/2))
            begin
              result = (num_1_tens*10 + num_1_units) / (num_2_tens * 10 + num_2_units) + 1;
            end
          else
            begin
              result = (num_1_tens*10 + num_1_units) / (num_2_tens * 10 + num_2_units);
            end
          div_error = 0;
        end
      else
        begin
          div_error = 1;
        end

        is_neg_thousands = 0;
        is_neg_tens = 0;
    end
end
```

```
always@(*)
begin
 if(is_neg_tens == 0 && is_neg_thousands == 0 && div_error == 0)
   begin
     result_units = target ? (result%10) : num_2_units;
     result_tens = target ? ((result/10)%10) : num_2_tens;
     result_hundreds = target ? ((result/100)%10) : num_1_units;
     result_thousands = target ? ((result/1000)%10) : num_1_tens;
   end
 else if(is_neg_tens == 0 && is_neg_thousands == 1 && div_error == 0)
   begin
     result_units = target ? (result%10) : num_2_units;
     result_tens = target ? ((result/10)%10) : num_2_tens;
     result_hundreds = target ? 10 : num_1_units;
     result_thousands = target ? 11 : num_1_tens;
   end
 else if(is_neg_tens == 1 && is_neg_thousands == 0 && div_error == 0)
   begin
     result_units = target ? (result%10) : num_2_units;
     result_tens = target ? 10 : num_2_tens;
     result_hundreds = target ? 11 : num_1_units;
     result_thousands = target ? 11 : num_1_tens;
   end
 else if(div_error == 1)
   begin
     result_units = target ? 13 : num_2_units;
     result_tens = target ? 12 : num_2_tens;
     result_hundreds = target ? 13 : num_1_units;
     result_thousands = target ? 14 : num_1_tens;
```

```
    end
end


always @(posedge B1, posedge B2, posedge B3, posedge B4, posedge B5, posedge B6, posedge B7,
posedge B8, posedge B9)
begin
  if (B1 || B2 || B3 || B4 || B9)
    target = 0;
  else if (B5 || B6 || B7 || B8)
    target = 1;
  end
endmodule
```