```
module lz77_decompressor(
  input wire clk,
  input wire reset,
  input wire start,
  input wire [7:0] token_offset,
  input wire [7:0] token_length,
  input wire [7:0] token_codeword,
  output reg [7:0] decompressed_data,
  output reg data_valid,
  output reg done
);

  parameter MAX_STRING_LENGTH = 256;

  reg [7:0] buffer [0:MAX_STRING_LENGTH-1];
  reg [7:0] buffer_pos;
  reg [7:0] output_pos;
  reg [7:0] match_pos;
  reg [7:0] match_length;
  reg processing;
  reg copying;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      buffer_pos <= 0;
      output_pos <= 0;
      match_pos <= 0;
      match_length <= 0;
      processing <= 0;
      copying <= 0;
      data_valid <= 0;
```

```
        done <= 0;
    end else if (start) begin
        processing <= 1;
        copying <= 0;
        match_pos <= buffer_pos - token_offset;
        match_length <= token_length;
    end else if (processing) begin
        if (token_offset == 0 && token_length == 0) begin
            // Literal character
            buffer[buffer_pos] <= token_codeword;
            decompressed_data <= token_codeword;
            data_valid <= 1;
            buffer_pos <= buffer_pos + 1;
            output_pos <= output_pos + 1;
            processing <= 0;
        end else if (!copying && match_length > 0) begin
            // Start copying matched data
            copying <= 1;
        end else if (copying && match_length > 0) begin
            // Continue copying matched data
            buffer[buffer_pos] <= buffer[match_pos];
            decompressed_data <= buffer[match_pos];
            data_valid <= 1;
            buffer_pos <= buffer_pos + 1;
            output_pos <= output_pos + 1;
            match_pos <= match_pos + 1;
            match_length <= match_length - 1;
        end else if (copying && match_length == 0) begin
            // Finish copying and add codeword
            buffer[buffer_pos] <= token_codeword;
            decompressed_data <= token_codeword;
            data_valid <= 1;
            buffer_pos <= buffer_pos + 1;
            output_pos <= output_pos + 1;
            copying <= 0;
            processing <= 0;
          end
        end else begin
          data_valid <= 0;
          if (output_pos == MAX_STRING_LENGTH) begin
            done <= 1;
          end
        end
      end

endmodule
```