Black Box Diagram

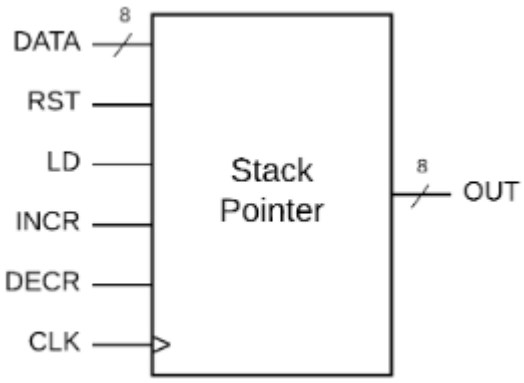

Figure 1: Stack Pointer Black Box Diagram

Behavior Description

The Stack Pointer is a hardware module. It keeps track of the address in the Scratch RAM that is being occupied by the top of the stack. It is used as an address input into the Scratch RAM for pushing data onto and popping data off of the stack. It keeps track of where the top of the stack is so the programmer does not have to keep track in the assembly code. As data is pushed on top of the stack, the address is decremented, and as data is popped off the top of the stack, the address is incremented. The stack pointer can also be reset or loaded with an arbitrary value. The Scratch RAM is used for the dual purpose of accessing data in arbitrary locations (ST and LD) and from the stack (PUSH and POP). The stack is also used when making subroutine calls. It is used to save the current address in the Program Counter for a CALL instruction. The RET instruction will loads the Program Counter with the address saved from the stack.

Structural Design

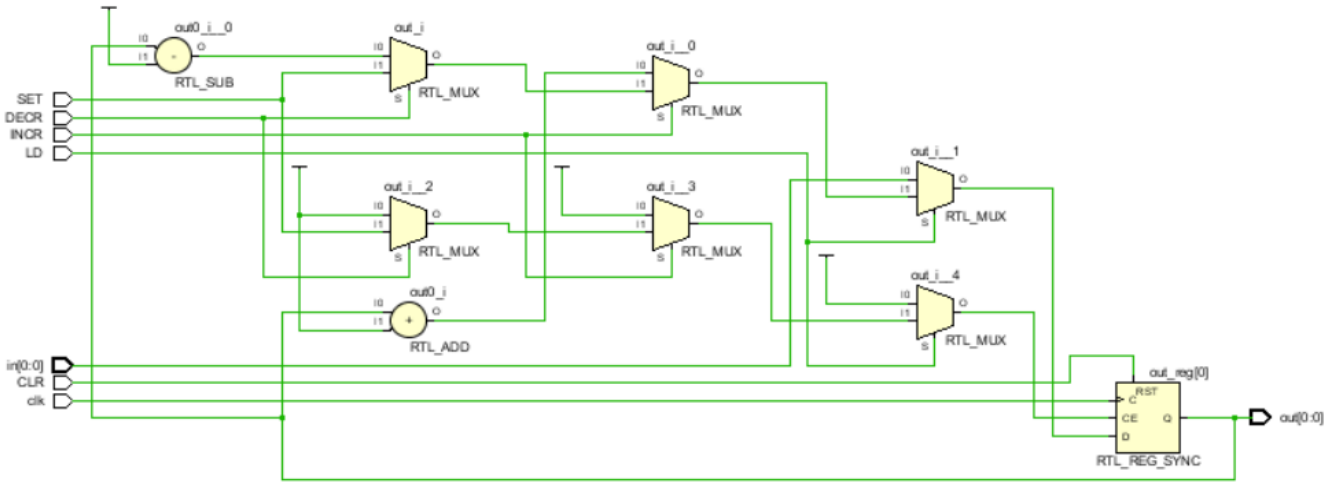

Figure 2: Stack Pointer Structural Design

System Verilog Code

Stack Pointer – (used the General Register)

```
//////////////////////////////////////////////////////////////////
// Engineer: Marina Dushenko and Matt Bailey
// Create Date: 02/11/2019 09:35:28 AM
// Module Name: FLAG_REGISTER
// Description: Register for flags, but can be used
// for other things
// Modifications: this file was modified to act as a general
// register on 2/23/2019 - Marina
//////////////////////////////////////////////////////////////////
`timescale 1ns / 1ps

module GeneralRegister #(parameter WIDTH = 1)(

input clk,

input [WIDTH - 1 : 0] in,

input SET, LD, CLR, DECR, INCR,

output logic [WIDTH - 1 : 0] out );


 always_ff @ (posedge clk)

 begin

   if (CLR) out = 0;

   else if (LD) out = in;

   else if (INCR) out = out +1;

   else if (DECR) out = out -1;

   else if (SET) out = 1;

    end

endmodule
```

Control Unit

```
//////////////////////////////////////////////
// Engineer: Marina Dushenko and Matt Bailey
// Create Date: 02/05/2019 09:35:28 AM
// Module Name: CONTROL_UNIT
// Description: Brain of the RAT_CPU
// tells every other component what to do
// based on opcode
// FSM
// File was updated on 2/23/19 for RAT#7 - Marina
//////////////////////////////////////////////

`timescale 1ns / 1ps
module ControlUnit(
        input C,Z,INTERRUPT, RESET, CLK,
        input [4:0] OPCODE_HI_5,
        input [1:0] OPCODE_LOW_2,

           output logic PC_LD, PC_INC,
           output logic [1:0] PC_MUX_SEL,
           output logic ALU_OPY_SEL,
           output logic [3:0] ALU_SEL,
           output logic RF_WR,
           output logic [1:0] RF_WR_SEL,
           output logic FLG_C_SET, FLG_C_CLR, FLG_C_LD, FLG_Z_LD, RST,
IO_STRB,
           output logic SCR_DATA_SEL, SCR_WE, SP_DECR, SP_INCR, SP_LD,
           output logic [1:0]SCR_ADDR_SEL
                  );

   typedef enum {ST_INIT, ST_FETCH, ST_EXEC} STATE;
   STATE NS, PS = ST_INIT;
   logic [6:0] opcode;
   assign opcode = {OPCODE_HI_5, OPCODE_LOW_2};

   always_ff @ (posedge CLK)
    begin
          if (RESET == 1) PS <= ST_INIT;
          else            PS <= NS;
    end

   always_comb
    begin
         PC_LD = 0;  PC_INC = 0; PC_MUX_SEL = 0; ALU_OPY_SEL = 0;
         ALU_SEL = 0;  RF_WR = 0; RF_WR_SEL = 0; FLG_C_SET = 0; FLG_C_CLR =
0;
         FLG_C_LD = 0;  FLG_Z_LD = 0;  RST = 0;  IO_STRB = 0;
         SCR_DATA_SEL = 0; SCR_WE = 0; SCR_ADDR_SEL = 0;
         SP_DECR = 0; SP_INCR=0; SP_LD = 0;

   begin
     case (PS)
     ST_INIT:
          begin
                RST = 1;  NS = ST_FETCH;
```

```
          end
  ST_FETCH:
          begin
                PC_INC = 1;  NS = ST_EXEC;
          end
   ST_EXEC:
           begin
            case(opcode)
 //IN
              7'b1100100, 7'b1100101, 7'b1100110, 7'b1100111:
                 begin
                   RF_WR_SEL = 3; RF_WR = 1;
                 end
//MOV
       7'b0001001: //reg to reg case
        begin
             ALU_OPY_SEL = 0; ALU_SEL = 4'b1110; RF_WR_SEL = 2'b00;  RF_WR =
1;
         end
      7'b1101100, 7'b1101101, 7'b1101110, 7'b1101111:
                 begin
                   ALU_OPY_SEL = 1; ALU_SEL = 4'b1110;  RF_WR_SEL = 0;  RF_WR
= 1;
                 end
//EXOR
               7'b0000010: //reg to reg case
                begin
                  ALU_OPY_SEL = 0; ALU_SEL = 4'b0111;  FLG_C_CLR = 1; FLG_Z_LD
= 1; RF_WR_SEL = 0; RF_WR = 1;
                     end
             7'b1001000, 7'b1001001, 7'b1001010, 7'b1001011:
                 begin
                   ALU_OPY_SEL = 1; ALU_SEL = 4'b0111;   FLG_C_CLR = 1;
FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 1;
                 end
 //OUT
               7'b1101000, 7'b1101001, 7'b1101010, 7'b1101011:
               begin
                 IO_STRB=1;
               end
//BRN
               7'b0010000:
                 begin
                   PC_LD = 1; PC_MUX_SEL = 0;
                 end
//AND
               7'b0000000: // reg to reg case
                 begin
                   ALU_OPY_SEL = 0;                   ALU_SEL = 4'b0101;
FLG_C_CLR = 1; FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 1;
                 end
               7'b1000000, 7'b1000001, 7'b1000010, 7'b1000011:
                 begin
                   ALU_OPY_SEL = 1; ALU_SEL = 4'b0101; FLG_C_CLR = 1; FLG_Z_LD
= 1;  RF_WR_SEL = 0;  RF_WR = 1;
                 end
//OR
```

```
              7'b0000001: // reg to reg case
                begin
                  ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b0110;
                  FLG_C_CLR = 1; FLG_Z_LD = 1;  RF_WR_SEL = 0;  RF_WR = 1;
                end
              7'b1000100, 7'b1000101, 7'b1000110, 7'b1000111:
                begin
                  ALU_OPY_SEL = 1; ALU_SEL = 4'b0110;  FLG_C_CLR = 1;
FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 1;
                end
//TEST
              7'b0000011: // reg to reg case
                begin
                  ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b1000; FLG_C_CLR = 1;
FLG_Z_LD = 1; RF_WR_SEL = 2'b00; RF_WR = 0;
                end
              7'b1001100, 7'b1001101, 7'b1001110, 7'b1001111:
                begin
                  ALU_OPY_SEL = 1'b1; ALU_SEL = 4'b1000;  FLG_C_CLR = 1;
FLG_Z_LD = 1;  RF_WR_SEL = 2'b00; RF_WR = 0;
                end
//ADD
              7'b0000100: // reg to reg case
                begin
                  ALU_OPY_SEL = 0; ALU_SEL = 4'b0000; FLG_C_LD = 1;  FLG_Z_LD
= 1;  RF_WR_SEL = 0 ;RF_WR = 1;
                end
              7'b1010000, 7'b1010001, 7'b1010010, 7'b1010011:
                begin
                  ALU_OPY_SEL = 1; ALU_SEL = 4'b0000; FLG_C_LD = 1; FLG_Z_LD
= 1; RF_WR_SEL = 0; RF_WR = 1;
                end
//ADDC
              7'b0000101: // reg to reg case
                begin
                  ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b0001; FLG_C_LD = 1;
FLG_Z_LD = 1; RF_WR_SEL = 0;  RF_WR = 1;
                end
              7'b1010100, 7'b1010101, 7'b1010110, 7'b1010111:
                begin
                  ALU_OPY_SEL = 1; ALU_SEL = 4'b0001; FLG_C_LD = 1;  FLG_Z_LD
= 1;  RF_WR_SEL = 0; RF_WR = 1;
                end
 //SUB
              7'b0000110: // reg to reg case
                begin
                  ALU_OPY_SEL = 0; ALU_SEL = 4'b0010;
                  FLG_C_LD = 1; FLG_Z_LD = 1;  RF_WR_SEL = 0;  RF_WR = 1;
                end
              7'b1011000, 7'b1011001, 7'b1011010, 7'b1011011:
                begin
                  ALU_OPY_SEL = 1'b1;   ALU_SEL = 4'b0010;  FLG_C_LD = 1;
FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR = 1;
                end
//SUBC
              7'b0000111: // reg to reg case
                begin
```

```
                    ALU_OPY_SEL = 1'b0;  ALU_SEL = 4'b0011;  FLG_C_LD =
1;FLG_Z_LD = 1;  RF_WR_SEL = 0 ;RF_WR = 1;
                  end
                7'b1011100, 7'b1011101, 7'b1011110, 7'b1011111:
                  begin
                    ALU_OPY_SEL = 1;  ALU_SEL = 4'b0011; FLG_C_LD = 1;
FLG_Z_LD = 1; RF_WR_SEL = 0;   RF_WR = 1;
                  end
 //CMP
                7'b0001000: // reg to reg case
                  begin
                    ALU_OPY_SEL = 1'b0;  ALU_SEL = 4'b0100;   FLG_C_LD = 1;
FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 0;
                  end
                7'b1100000, 7'b1100001, 7'b1100010, 7'b1100011:
                  begin
                    ALU_OPY_SEL = 1'b1; ALU_SEL = 4'b0100;  FLG_C_LD = 1;
FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR = 0;
                  end
  //LD
                7'b0001010: // reg to reg case
                  begin
                  RF_WR = 1; RF_WR_SEL = 1; SCR_ADDR_SEL=0;
                  end
                7'b1110000, 7'b1110001, 7'b1110010, 7'b1110011:
                  begin
                  RF_WR = 1; RF_WR_SEL = 1; SCR_ADDR_SEL = 1;
                  end
//ST
                7'b0001011: // reg to reg case
                  begin
                  SCR_DATA_SEL = 0; SCR_WE = 1; SCR_ADDR_SEL = 0;
                  end
                7'b1110100, 7'b1110101, 7'b1110110, 7'b1110111:
                  begin
                   SCR_DATA_SEL = 0; SCR_WE = 1; SCR_ADDR_SEL = 1;
                  end
 //CALL
                7'b0010001:
                  begin
                  PC_LD = 1; SCR_DATA_SEL = 1; SCR_WE = 1; SCR_ADDR_SEL = 3;
SP_DECR = 1; PC_MUX_SEL=0;
                  end
 //BREQ
                7'b0010010:
                  begin
                    if (Z==1)   PC_LD = 1;
                    else        PC_LD = 0;
                  end
 //BRNE
                7'b0010011:
                  begin
                    if (Z==0) PC_LD = 1;
                    else      PC_LD = 0;
                  end
  //BRCS
                7'b0010100:
```

```
                begin
                  if (C==1)   PC_LD = 1;
                  else        PC_LD = 0;
                end
//BRCC
              7'b0010101:
                begin
                  if (C==0) PC_LD = 1;
                  else      PC_LD = 0;
                end
 //LSL
              7'b0100000:
                begin
                  ALU_SEL = 4'b1001; FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR = 1;
                end
            //LSR
              7'b0100001:
                begin
                  ALU_SEL = 4'b1010;   FLG_Z_LD = 1; RF_WR_SEL = 2'b00; RF_WR
= 1;
                end
    //ROL
              7'b0100010:
                begin
                  ALU_SEL = 4'b1011;  FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR =
1;
                end
            //ROR
              7'b0100011:
                begin
                  ALU_SEL = 4'b1100;  FLG_Z_LD = 1; RF_WR_SEL = 2'b00; RF_WR
= 1;
                end
            //ASR
              7'b0100100:
                begin
                  ALU_SEL = 4'b1101; FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 1;
                end
 //PUSH
              7'b0100101:
                begin
                 SCR_ADDR_SEL = 3; SCR_WE = 1; SCR_DATA_SEL = 0; SP_DECR = 1;
                end
 //POP
              7'b0100110:
                begin
                SCR_ADDR_SEL = 2; RF_WR_SEL = 1; RF_WR  = 1; SP_INCR = 1;
                end
            //WSP
              7'b0101000:
                begin
                SP_LD = 1;
                end
            //RSP
              7'b0101001:
                begin
                RF_WR_SEL = 2; RF_WR = 1;
```

```
                end
            //CLC
              7'b0110000:
                begin
                  FLG_C_CLR = 1;
                end
            //SEC
              7'b0110001:
                begin
                  FLG_C_SET = 1;
                end
            //RET
              7'b0110010:
                begin
                   SCR_ADDR_SEL = 2; PC_MUX_SEL = 1; PC_LD = 1; SP_INCR = 1;

                end
            //SEI
              7'b0110100:
                begin
                //FILL ME
                end
            //CLI
              7'b0110101:
                begin
                //FILL ME
                end
            //RETID
              7'b0110110:
                begin
                // FILL ME
                end
            //RETIE
              7'b0110111:
                begin
                //FILL ME
                end
           default: RST = 1;
         endcase
        NS = ST_FETCH;
        end
        default : NS = ST_INIT;
        endcase
       end
    end

endmodule
```