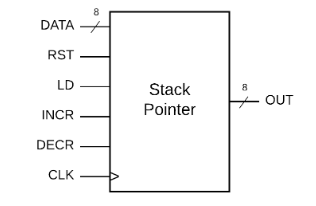
Black Box Diagram

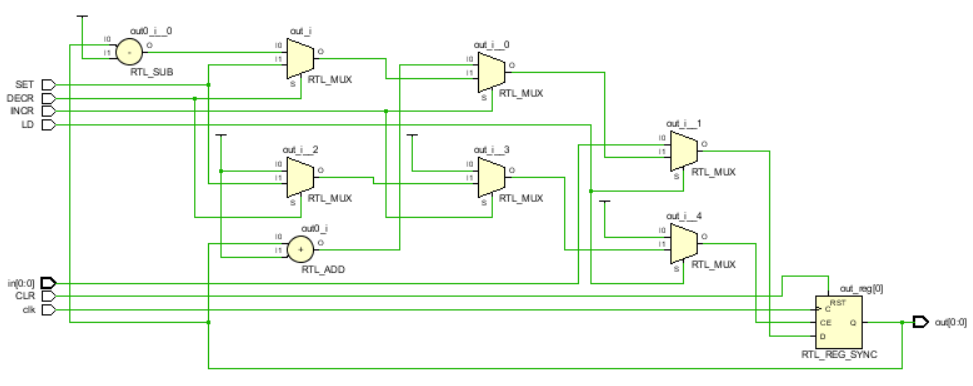
﻿﻿

*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