**Stack Based Calculator Documentation**

**Block Diagram**

**BUS**

**n-reg /tribuf**

**n-reg /tribuf**

**n-reg /tribuf**

**n-reg /tribuf**

**Mem Test**

**Mem Control**

**ADDER/SUB**

**Multiply**

**Divider**

**STACK CONTROL**

**Memory Control Unit Commands**

ENTITY Mem\_Control IS

PORT ( Clk,Rst : IN STD\_LOGIC;

CNTRL : IN STD\_LOGIC\_VECTOR (2 DOWNTO 0);

LIV,Z1, Z2 : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Bus\_In : INOUT STD\_LOGIC\_VECTOR(3 DOWNTO 0);

STATE : OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);

R : BUFFER STD\_LOGIC)

**Operations:**

Format: CNTRL| RI/RO | RI/RO

-----------------------

Format: 001 | z1,z1,z1,z1, | xxxx | R

Description: move LIV to register Z1

-----------------------

Format: 010 | z1,z1,z1,z1 | z2,z2,z2,z2 | R

Description: Move Z1 register into Z2 register

-----------------------

011 | z1,z1,z1,z1 | xxxx | R

MOV Z TO BUS

-----------------------

100 | z1,z1,z1,z1 | xxxx | R

MOV BUS INTO Z1

-----------------------

101 | xxxx | xxxx | R

CLR REGs

-----------------------

Notes: R = done bit , State = temp debug , a RST pulse must occur before a command will execute. Command bits must be present until R done bit is asserted.

Assignment:

ECE 375 Project: Stack-Based Calculator

Design a stack-based calculator for 8-bit signed binary numbers. Input data

to the calculator is from the DIP switches with a separate push-button to enter

the data. The calculator should have the following operations:

– enter: push the 8-bit input data onto the stack

– 0 - clear: clear the top of the stack, reset the stack counter, reset overflow,

and so on.

– 1 - add: replace the top two data entries on the stack with their sum.

– 2 - sub: replace the top two data entries on the stack with their difference

(stack top - next entry).

– 3 - mul: replace the top two data entries on the stack with their product (8

bits X 8 bits to give 8-bit product).

– 4 - div: replace the top two data entries on the stack with their quotient

(stack top / next entry) (8 bits divided by 8 bits to give 8-bit quotient).

– 5 - xchg exchange the top two data entries on the stack.

– 6 - neg replace the top of the stack with its 2’s complement.

Negative numbers should be represented in 2’s complement. Provide an over-

flow indicator for 2’s complement overflow. This indicator should always be set

if the product requires more than 8 bits including sign or if divide by 0 is at-

tempted.

Implement a stack module that has four 8-bit words. The stack should have

the following operations: push, pop, and exchange the top two words on the stack.

The top of the stack should always be displayed on the eight LEDs. Include an

indicator for stack overflow (attempt to push a fifth word) and stack underflow

(attempt to pop an empty stack or to exchange the top of the stack with an

empty location).

Design the control unit for the calculator using linked state machines. Draw

a main SM chart with separate SM charts for the multiplier and divider control.

When you design the arithmetic unit, try to avoid adding unnecessary registers.

You should be able to implement the arithmetic unit with three registers (8 or

9 bits each), an adder, two complements, and so on.