Homework 5: Stack Machine

The Stack Machine

For this assignment you will construct the instruction decoder for a stack machine. The stack machine has 16 registers organized as a stack. These 4-bit registers are numbered st00 through st15. As in assembly language programming, the bottom of the stack has the larger address. That is, the bottom of the stack is st15 and the top of the stack grows towards st00. The stack has a counter that keeps track of the register that is the current top of the stack.

During each cycle, the top two items of the stack are available simultaneously. In this design, the top of the stack is routed directly to Port A of the ALU and the register just below the top of the stack is routed to Port B of the ALU.

The stack counter can be instructed to either decrement (e.g., to PUSH), increment (to POP), stay the same, or reset to 0, according to the following:

sc1	sc0	effect
0	0	reset to zero
0	1	increment (POP)
1	0	no change
1	1	decrement (PUSH)

The contents of the stack can be modified using the 4-bit wide WRITE line. The data on the WRITE line can be written to either the current top of the stack, the register above it or the register below it. The register being modified is specified by the write select lines (WSEL):

wsel1	wsel0	effect
0	0	write to current top
0	1	write to register below
1	0	no write
1	1	write to register above

The WRITE line is connected to the output of a 4-to-1 multiplexer (MUX). This multiplexer chooses one of: the output of the ALU, the top of the stack or the immediate value embedded in the stack machine's instruction (see below). The fourth input line is not used:

MUX 1	MUX0	input selected
0	0	output of ALU
0	1	top of the stack
1	0	immediate operand
1	1	not used

The ALU in this circuit is very similar to the one in Homework 3. The one exception is that the subtraction instruction subtracts Port A from Port B, rather than vice versa. (Also, the circuitry is implemented using addition and subtraction modules in Logisim instead of gates.) As in Homework 3, the ALU function is selected via two control lines, d1 and d0:

d1	d0	effect
0	0	add Port A and Port B
0	1	subtract Port A from Port B
1	0	decrement Port A
1	1	increment Port A

Finally, this stack machine has an "output" of 4 LED's. When the LED's control line is set to 1, the 4-bit value at the top of the stack is displayed.

The Instruction Set

With these controls in place, we can construct an instruction set for our stack machine. For example, we can implement an ADD instruction that pops the top two items off the stack and places their sum at the top of the stack. To accomplish this task, we can set the following values for the control lines:

MUX1	MUX0	= 0 0	output of ALU
sc1	sc0	= 0 1	add 1 (pop)
wsel1	wsel0	= 0 1	write to register below top of stack
d1	d0	= 0 0	tell ALU to ADD
	LED	= 0	don't display

For example, if the current top of the stack is register st05 and the registers st05 and st06 hold 9 and 4 (respectively), then after the ADD instruction, the top of the stack is register st06 and st06 will hold the value 13. (The st05 register will still hold the value 9, but that does not matter since we do not need to actually erase values that are popped off the stack.)

Your assignment is to implement the following instructions for our stack machine:

ADD Pop the top two items off the top of the stack and push their sum.
The resulting stack has one less item.

SUB Pop the top two items off the top of the stack and push their difference.
The resulting stack has one less item.

DEC Reduce the value at the top of the stack by 1.
The resulting stack has the same number of items.

INC Increase the value at the top of the stack by 1.
The resulting stack has the same number of items.

DUP Read the top of the stack and push this value on the stack. The top of the stack has been "duplicated". The resulting stack has one more item.

MOV Store the given immediate value in the top of the stack.
The resulting stack has the same number of items.

PUSH Push the given immediate value in the top of the stack.
The resulting stack has one more item.

DROP Remove the top of the stack.
The resulting stack has one less item.

SHOW Display the value in the top of the stack on the LED's. This is the only instruction that uses the LED's. The LED should be off in all other instructions.
The resulting stack has the same number of items.

RESET Store 0 in the stack counter.

The op code for each instruction is given in the table on the next page. Each instruction takes 8 bits. The most significant 4 bits (i7, i6, i5, i4) specifies the instruction. The remaining 4 bits (i3 i2 i1 i0) specify the immediate operand. Only two instructions use the immediate operand: PUSH and MOV. For example, the op code for "PUSH 3" is 1010 0011 = A3.

Your Assignment

Your assignment is to complete the instruction table on the next page by specifying the values that must be placed on the control lines to accomplish the remaining instructions (ADD is already done for you). Next, use the Karnaugh maps provided to simplify the Boolean formula for each control line. *Clearly indicate your Boolean formula for each control line*.

Finally, implement the resulting circuit in Logisim. You just need to add circuitry to the "Instruction Decoder" module in the given circuit

What to submit

In class on Tuesday May 14, turn in your truth table and Karnaugh maps on paper. Make copies of these, if you still need them to implement your circuit in Logisim.

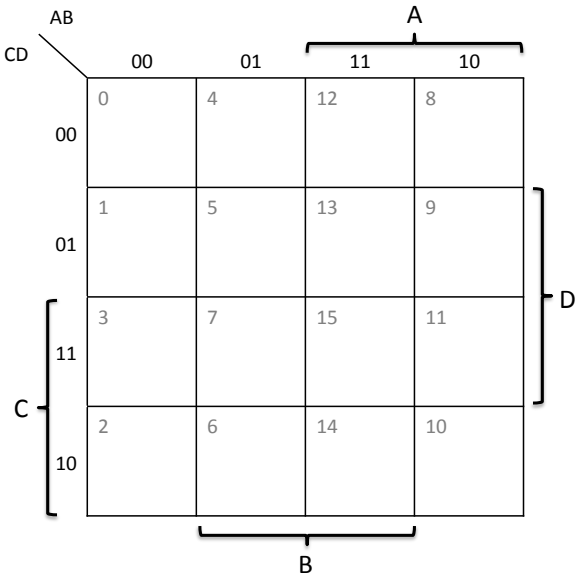
In Logisim, save the circuit file, then transfer the file to your account on GL. Finally, submit the circuit file using the submit command. The name of this assignment is hw5, so your submit command should look something like:

```
submit cmsc313 hw5 hw5.circ
```

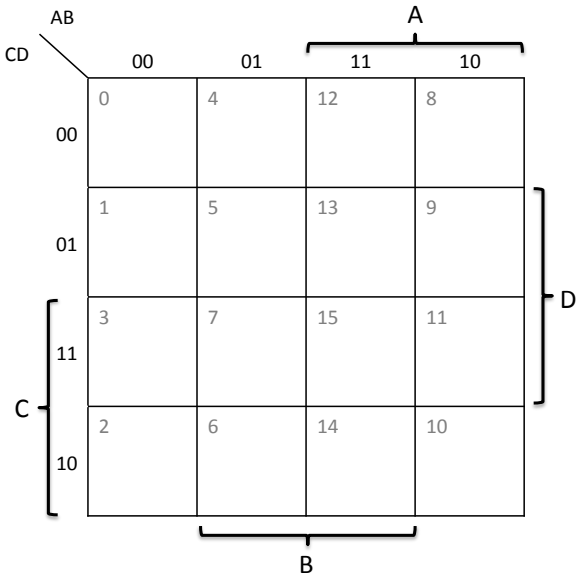
CMSC 313 Homework 5, Truth Table

	i7	i6	i5	i4	MUX 1	MUX 0	SC 1	SC 0	WSEL 1	WSEL 0	ALU d1	ALU d0	LED
ADD	0	0	0	0									
SUB	0	0	0	1									
DEC	0	0	1	0									
INC	0	0	1	1									
X	0	1	0	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
X	0	1	0	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
X	0	1	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
X	0	1	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
DUP	1	0	0	0									
MOV	1	0	0	1									
PUSH	1	0	1	0									
DROP	1	0	1	1									
SHOW	1	1	0	0									
RESET	1	1	0	1									
X	1	1	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
X	1	1	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*

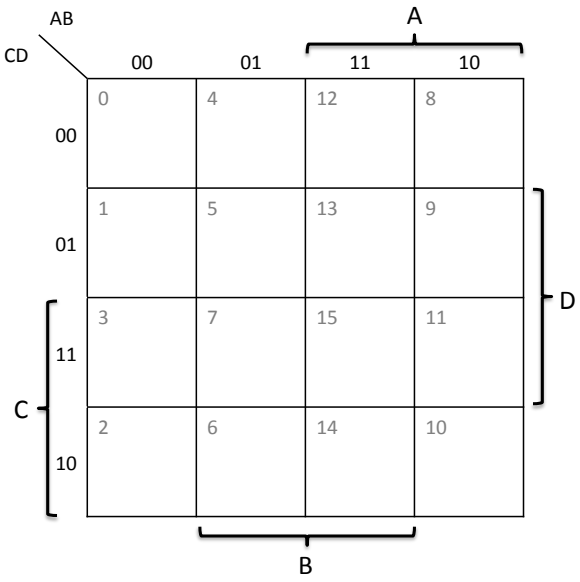
MUX 1



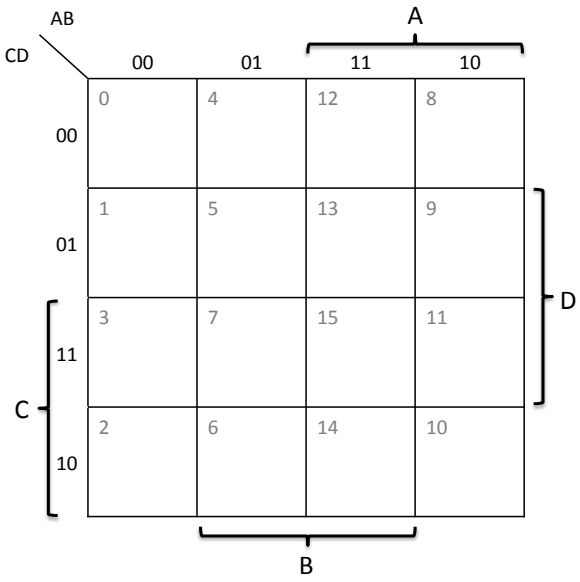
MUX 0



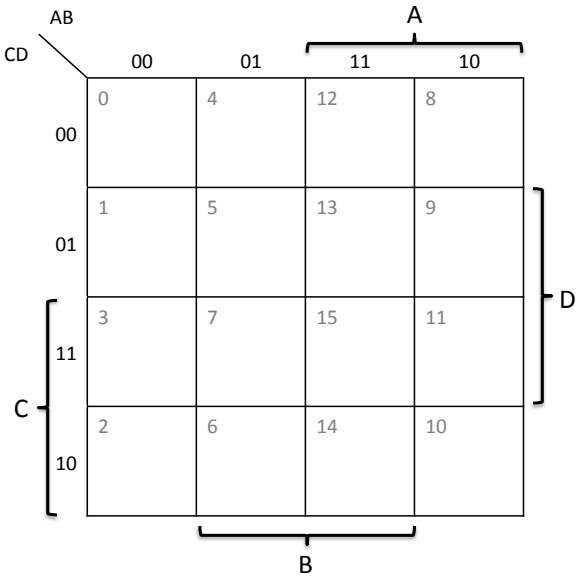
SC1



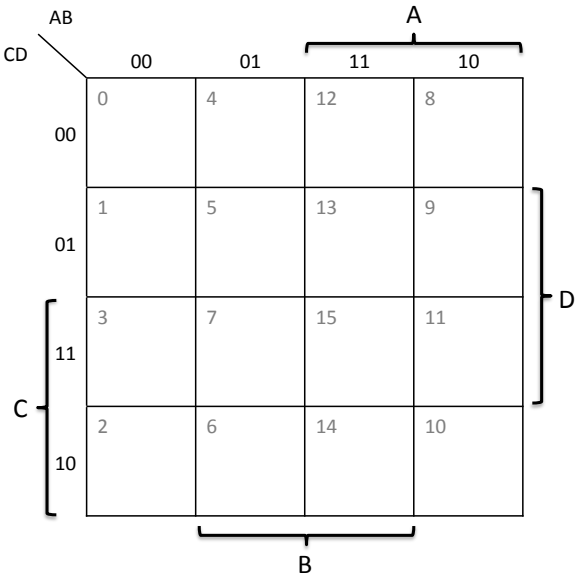
SC0



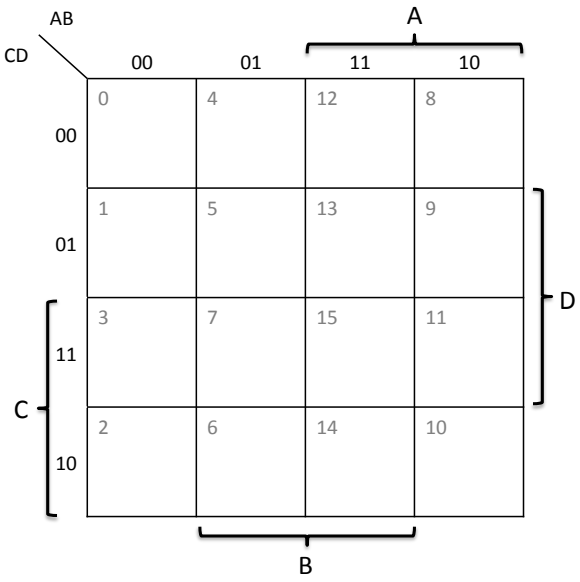
WSEL 1



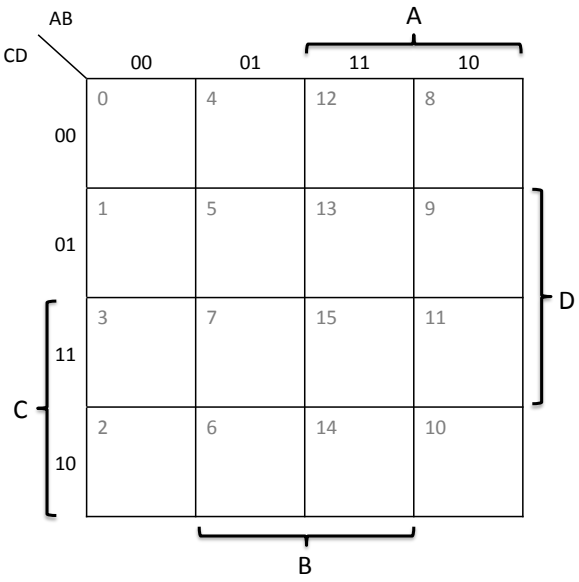
WSEL 0



ALU d1



ALU d0



LED