Pseudocode for April

The purpose of this document is to help April think about how to incorporate the switches and buttons into the SDES design.

The first thing you're gonna want to do is instantiate an Encrypt module, a Decrypt module, and a KeyGen module.

Encrypt:
Inputs: Plaintext, Subkey1, Subkey2
Output: Ciphertext

Decrypt:
Inputs: Ciphertext, Subkey1, Subkey2
Output: Plaintext

KeyGen:
Inputs: 10 bit key
Outputs: Subkey1, and subkey2

This design could use two buttons to load up either a message (to be encrypted or decrypted), or the 10 bit key key, 10 LED lights to display the 10 bit key, 8 hex keys to display the message, the subkeys and the result of either encryption or decryption. It should also include 10 switches used to load the 10 bit key in, and 1 switch to control whether we are encrypting or decrypting. (We could also reuse the same switches that we use to load up the 10 bit key, to load up the 8 bit message (to be encrypted or decrypted).

Example pseudocode design:

High level/simple:
KeyGen instance(pass in inputs and outputs)

More detailed/complex:
KeyGen instance(10bitkeyreg, subreg1, subreg2);

The subkeys should be saved in 8 bit registers, we could call them subreg1 and subreg2. The 10 bit key must be passed in. One way to do this is maybe when we press a button, whatever configuration the 10 switches are in, maybe that gets saved in a

register (which we could call 10bitkeyreg). Then this register is loaded up onto the LEDs and whatever values are in subreg1 and subreg2, those values would appear on the seven segment display.

In verilog, I would just do a simple assign statement. Whatever value is in the 10bitkeyreg, just wire that to the LEDS.

Example:
```
Assign LEDR[9:0] = 10bitkeyreg;  //I'm pretty sure this would work but this has not been
                                                    // tested!
```

You may be wondering how we even get the input from the switches into 10bitkeyreg. In verilog we use something called an always block. It looks like this:

```
always@(posedge of KeyLoadingButton) begin
        10bitkeyreg = SW[9:0];
End
```

And its as simple as that. Whenever the button is pushed, the value on the switches is loaded into the register.

Moving on...

High level/simple:
Encrypt instance1(pass in inputs and outputs)

More detailed/complex
Encrypt instance1(MessageReg, Subreg1, Subreg2, cryptoreg);

Plaintext is saved in an 8 bit register, lets call it MessageReg. Subreg1 and subreg2 are both passed into the encrypt instance. The output is the cypertext and that could be saved in a register and then passed into a seven segment module to be displayed in Hex. We could call this register cryptoreg

However, we should not display the cipher text until the encrypt button has been pushed.

In verilog we would do this with an always block.

```
always@(posedge of Encryptbutton) begin
        Hex0 = cryptoreg; //here we are loading the cryptoreg into the hex display.
End
```

This means that when the Encryptbutton is pushed, the code inside “begin” and “end” executes. Its kind of like a clocked register.

*Note
The register for the cyphertext (cryptoreg) can be automatically loaded up without any button. All you have to do is tie cryptoreg to the output port in the Encrypt instance1.

We could make this more complex

```
always@(posedge of Encryptbutton) begin
        If (Encrypt)
        {
                Hex0 = cryptoreg;
        }
        Else if (Decrypt)
                Hex0 = DecryptedMessage;
End
```

You could make another block like this one for a different button:

```
always@(posedge of LoadMessageButton) begin
        MessageReg = plaintext; //load the input on the switches into a register

        Hex1 = MessageReg; //load seven segment display with plain text
End
```

I want to emphasize that the 10 switches can be used for more than one purpose. They don’t have to JUST be used for inputting the 10 bit key. We could use their input for entering the message we either want to encrypt or decrypt as well. The way we add extra functionality is by using the buttons.The buttons decide where the data goes.

A good idea:
One more block just like the above could be made to load in a ciphertext (to be decrypted). Just make it so that it gets loaded into a register and into a seven seg

display when the load ciphertext button is pushed. Again this functionality is for when we want to decode a message.

A better idea:
Alternatively, we could use the same button and hex display for both the plaintext and the ciphertext(to be decrypted). In this way we would just treat this as an all purpose “message” register. Whatever is in that register can be either encrypted or decrypted depending on what button you push. I hope that makes sense.