Saul Reyna
EECE 2160

Lab 3 Pre-lab

Assignment 1

What `RegisterWrite(char *pBase, int offset, int value)` does is that it first gets pointer to the base address of the memory bus and add the an offset where the result will be a pointer to the address of whatever the function is writing to. This pointer is then assigned the value passed through the register function.

Then `RegisterRead(char *pBase, int offset)` passes that initial base address pointer, just as in the write function and adds the offset to the pointer. Then the function will return whatever value was written and is store in that new pointer.

The first line of the `*Initialize(int *fd)` function opens the memory for the device with the read/write flags. Then, mmap passes the `NULL` argument, letting the kernel choose the address to begin the mapping, along with the size, which sets the end of the mapping. Once the mapping is done, the protection flags for letting the memory be read and written to are passed and allows other memory maps that are mapping the same region to view any updates on the memory map. The memory map is then initialized using the physical address of all the I/O using the pointer to the file with the device's memory that was initially passed through the `Initialize` function.

The `Finalize(char *pBase, int fd)` unmaps the memory map that was initialized with the `Initialize` function by passing through all the addresses in the range of the base address of the map and the size of the mapping and deleting all the references. Once all the references are deleted, the file descriptor is closed, just as one would close an opened text file to avoid corruption.

Assignment 2

```
/** Changes the state of an LED (ON or OFF)
 * @param pBase     base address of I/O
 * @param ledNum    LED number (0 to 7)
 * @param state     State to change to (ON or OFF)
 */
void Write1Led(char *pBase, int ledNum, int state) {
    if(0 >= ledNum > 7) {
        std::cerr << "Trying to write to a non-existing LED" << std::endl;
    }

    int ledOffset = 0x12C + ((ledNum) * 4);

    RegisterWrite(pBase, ledOffset, state);
}
```

Saul Reyna
EECE 2160

Assignment 3

```
/** Reads the value of a switch
 * - Users base address of I/O
 * @param pBase     base address of I/O
 * @param switchNum Switch number (0 to 7)
 * @ return         Switch value read
 */
int Read1Switch(char *pBase, int switchNum) {
    if(0 >= ledNum > 7) {
        std::cerr << "Trying to read a non-existing switch" << std::endl;
    }

    int switchOffset = 0x14C + ((switchNum * 4);

    RegisterRead(pBase, switchOffset);
}
```