LC2K Simulator

Rijun Cai
12348003

January 4, 2014

In this laboratory, an LC2K simulator is implemented in C++. This is a brief introduction of my source code.

The main part of my program is the `LC2KMachine` class. The definition of it is as follow.

```
class LC2KMachine
{
    public:
        static const size_t NUMMEMORY = 65536;
        static const size_t NUMREGS = 8;

    private:
        size_t _mem_size, _pc;
        bool _ready;
        word_t _registers[NUMREGS];
        word_t *_memory;

        inline void handleAdd(mc_t mc);
        inline void handleNand(mc_t mc);
        inline void handleLw(mc_t mc);
        inline void handleSw(mc_t mc);
        inline void handleBeq(mc_t mc);
        inline void handleJalr(mc_t mc);
        inline void handleHalt();
        inline void handleNoop();

        int interpretOffset(mc_t mc)
        {
            mc_t offset = mc & 0xffff;
            return (offset & 0x8000) ? offset - 0x10000 : offset;
        }

    public:
        LC2KMachine(): _memory(new word_t[NUMMEMORY]), _ready(false) {}
        ~LC2KMachine()
        {
            if(_memory)
                delete [] _memory;
        }

        void next();
        void setMachineCode(std::vector<word_t> mc);
        void printState(std::ostream &os = std::cout);
        void printInitMem(std::ostream &os = std::cout);
};
```

The public function `LC2KMachine.setMachineCode()` is used to install an LC2K program into the simulator. It will write the program into the memory of the simulator.

`LC2KMachine.next()` function accepts no argument. It will execute the *current* instruction in the memory and move the program counter to the next instruction to be executed. It will get an instruction from the memory and decode it, then invoke the needed instruction handler functions (`LC2KMachine.handle*()`) to execute the instruction.

`LC2KMachine.printState()` and `LC2KMachine.printInitMem()` functions are used to output the state of the simulator.