labS 实验报告

李远航
PB20000137

1. 实验内容

- 学习使用CMAKE
- 阅读理解代码结构
- 正确填写`TO BE DONE`部分代码

2. 实验过程

(1) cmake的使用

进入项目目录

```
mkdir build
cd build
cmake ..
make
```

(2) 代码补全过程

- 程序结构
  - 命令行参数
    - `-h`&`--help` 菜单
    - `-f`&`--file` 输入文件
    - `-r`&`--register` 寄存器初始化
    - `-s`&`--single` single step
    - `-b`&`--begin` 开始地址
    - `-o`&`--output` 输出文件
    - `-d`&`-detail` 详细信息
  - 程序运行过程
    - 读取文件内容
    - 每次运行取PC当前的值，然后根据指令进行相关操作，每次运行使step数递增
  - TO BE DONE
    - Single step

```
if (!virtual_machine.NextStep())
    halt_flag =
if (gIsDetailedMode)
    std::cout << virtual_machine.reg << std::endl;
if (halt_flag)
    ++time_flag;
```

    - 读取文件内容部分

```
namespace virtual_machine_nsp
```

```
{
    void memory_tp::ReadMemoryFromFile(std::string filename,
int beginning_address)
    {
        std::ifstream infile(filename.c_str(), std::ios::in);
        if (!infile)
        {
            std::cout << "文件打开错误" << std::endl;
        }
        std::string ans;
        int index = 0;
        while (infile >> ans)
        {
            memory[beginning_address + index] =
TranslateInstruction(ans);
            index++;
        }
        infile.close();
    }

    int16_t memory_tp::GetContent(int address) const
    {
        return memory[address];
    }

    int16_t &memory_tp::operator[](int address)
    {
        return memory[address];
    }
}; // virtual machine namespace
```

- **inline T SignExtend(const T x)**

  符号位扩展

```
int select = (x >> (B - 1)) & 0b1;
if (!select)
    return x;
else
{
    int16_t temp = 0;
    for (int i = 15; i >= B; i--)
        temp += (1 << i);
    return x + temp;
}
```

- **void virtual_machine_tp::UpdateCondRegister(int regname)**

  更新条件码

```
int16_t temp = reg[regname];
if (temp == 0)
    reg[R_COND] = 0b010;
else if (temp & (1 << 15))
    reg[R_COND] = 0b100;
else
    reg[R_COND] = 0b001;
```

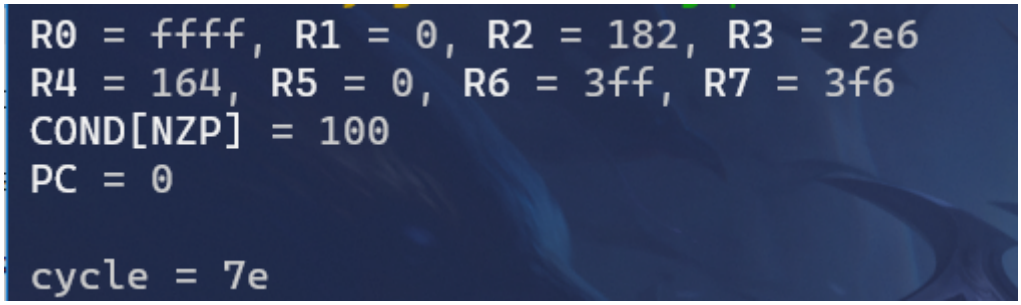
- 具体指令部分与LC-3执行的过程相同

```
void virtual_machine_tp::VM_AND(int16_t inst)
{
    int flag = inst & 0b100000;
    int dr = (inst >> 9) & 0x7;
    int sr1 = (inst >> 6) & 0x7;
    if (flag)
    {
        int16_t imm = SignExtend<int16_t, 5>(inst & 0b11111);
        reg[dr] = reg[sr1] & imm;
    }
    else
    {
        int sr2 = inst & 0x7;
        reg[dr] = reg[sr1] & reg[sr2];
    }
    UpdateCondRegister(dr);
}
void virtual_machine_tp::VM_JMP(int16_t inst)
{
    int16_t br = (inst >> 6) & 0b111;
    reg[R_PC] = reg[br];
}
void virtual_machine_tp::VM_JSR(int16_t inst)
{
    int select = (inst >> 11) & 0b1;
    reg[R_R7] = reg[R_PC];
    if (select)
    {
        reg[R_PC] = reg[R_PC] + SignExtend<int16_t, 11>(inst &
0x7FF);
    }
    else
    {
        int br = (inst >> 6) & 0x7;
        reg[R_PC] = reg[br];
    }
}
void virtual_machine_tp::VM_LDI(int16_t inst)
{
    int16_t dr = (inst >> 9) & 0x7;
    int16_t pc_offset = SignExtend<int16_t, 9>(inst & 0x1FF);
    reg[dr] = mem[mem[reg[R_PC] + pc_offset]];
    UpdateCondRegister(dr);
}
void virtual_machine_tp::VM_LDR(int16_t inst)
{
    int16_t dr = (inst >> 9) & 0x7;
    int16_t br = (inst >> 6) & 0x7;
    int16_t pc_offset = SignExtend<int16_t, 6>(inst & 0x3F);
    reg[dr] = mem[reg[br] + pc_offset];
    UpdateCondRegister(dr);
}
void virtual_machine_tp::VM_LEA(int16_t inst)
{
    int16_t pc_offset = SignExtend<int16_t, 9>(inst & 0x1FF);
```

```
55      int16_t dr = (inst >> 9) & 0x7;
56      reg[dr] = reg[R_PC] + pc_offset;
57  }
58  void virtual_machine_tp::VM_NOT(int16_t inst)
59  {
60      int16_t dr = (inst >> 9) & 0x7;
61      int16_t sr = (inst >> 6) & 0x7;
62      reg[dr] = ~reg[sr];
63      UpdateCondRegister(dr);
64  }
65  void virtual_machine_tp::VM_ST(int16_t inst)
66  {
67      int16_t sr = (inst >> 9) & 0x7;
68      int16_t pc_offset = SignExtend<int16_t, 9>(inst & 0x1FF);
69      mem[reg[R_PC] + pc_offset] = reg[sr];
70  }
71  void virtual_machine_tp::VM_STI(int16_t inst)
72  {
73      int16_t sr = (inst >> 9) & 0x7;
74      int16_t pc_offset = SignExtend<int16_t, 9>(inst & 0x1FF);
75      mem[mem[reg[R_PC] + pc_offset]] = reg[sr];
76  }
77  void virtual_machine_tp::VM_STR(int16_t inst)
78  {
79      int16_t sr = (inst >> 9) & 0x7;
80      int16_t br = (inst >> 6) & 0x7;
81      int16_t pc_offset = SignExtend<int16_t, 6>(inst & 0x3F);
82      mem[reg[br] + pc_offset] = reg[sr];
83  }
```

3. 程序运行效果

实例运行lab2程序，初始寄存器R0为15，理论答案R7为1014，即0x3f6



4. 实验收获

- 学习了cmake的使用
- 对boost库有了一定的了解
- 对LC-3指令有了更深的认识
- 提升了阅读代码的能力