LAB_A & LAB_S report

PB20151793 宋玮

LAB_A

实验目的和实验思路

程序的目标是实现一个小型汇编程序，其功能是转换汇编，将语言代码转换成二进制代码。让我们考虑一下两者之间的区别。在汇编语言中，除了二进制代码，我们可以使用伪指令、注释以及用labels签来替换PCoffset。然而，要实现我们的目标，我们必须考虑如何将这些转化回二进制文件。

由助教给出的框架的基本思路是两次pass：第一次扫描生成符号表；第二次扫描则完成将汇编语言转为二进制码的任务。

实验过程

1.转换功能

要实现汇编语言转为二进制编码，就需要用到一系列类型的转换，例如字符串转为10进制整形数，再将10进制数转为二进制码。又或者是将labels、寄存器、立即数转换为二进制码。

e.g.

```
std::string assembler::TranslateOprand(int current_address, std::string str,
int opcode_length) { //将labels、寄存器、立即数转换为二进制码
// Translate the oprand
    str = Trim(str);
    auto item = label_map.GetValue(str);
    if (!(item.getType() == vAddress && item.getVal() == -1)) {
    // str is a label
        auto label_address = item.getVal();
        auto temp = label_address - current_address - 1;
        auto s = NumberToAssemble(temp);
        return s.substr(16 - opcode_length, opcode_length);
    }
    if (str[0] == 'R') {
    // str is a register
        switch (str[1]) {
            case '0':return "000";
            case '1':return "001";
            case '2':return "010";
            case '3':return "011";
            case '4':return "100";
            case '5':return "101";
            case '6':return "110";
            case '7':return "111";
            default:;
        }
    }
    else {
    // str is an immediate number
    int inst = RecognizeNumberValue(str);
    std::string s = NumberToAssemble(inst);
    return s.substr(16 - opcode_length, opcode_length);
```

```
    }
}

```

2.pass

第一次扫描是要讲代码与注释分离开来，故找到每行' '所在的位置，其前面的部分为代码，后面的部分为注释；

第二次扫描主要用来处理伪指令，标记每一行的指令类型，并记录每个标签对应的地址。我们应该遍历file_content, 使用line_address记录当前语句与起始语句的地址之间的偏移量，使用file_address记录PC相对于起始地址的偏移量。Label_mp记录标签对应的相对地址，方便计算偏移量的后续跳转语句。对于每一行，如果是注释，则不是处理。如果是伪指令，则需要在一定程度上完成。

在第三次扫描中，翻译结果被输出到文件中。对于指令类型的操作，我们只需要通过引用ISA输出它们。大致分为两个步骤:

1. result_line.append(opcode + nonvarible bits)
2. 判断该语句是否正确。如果正确，则通过TranslateOprand()函数将参数转换为二进制代码，并将它们写到result_line

e.g.

```
if (pseudo_command == ".FILL") {
    file_address[line_index] = line_address;
    std::string inst;
    line_stringstream >>inst;
    auto num_temp = RecognizeNumberValue(inst);
    if (num_temp == std::numeric_limits<int>::max()) {
        return -4;
    }
    if (num_temp > 65535 || num_temp < -65536) {
        return -5;
    }
}
if (IsLC3Command(word) != -1 || IsLC3TrapRoutine(word) != -1) {
// * This is an operation line
    file_tag[line_index] = lOperation;
    continue;
}
if (word == ".BLKW") {
// modify label map
// modify line address
    line_stringstream >> word;
    auto num_temp = RecognizeNumberValue(word);
    if (num_temp == std::numeric_limits<int>::max()) {
    // @ Error Invalid Number input @ BLKW
        return -11;
    }
    if (num_temp > 65535 || num_temp < -65536) {
    // @ Error Too large or too small value @ BLKW
        return -12;
    }
    label_map.AddLabel(label_name, value_tp(vAddress, line_address - 1));
    line_address += (num_temp - 1);
}
```

LAB_S

实验目的和实验思路

本实验是编写一个模拟器，实现机器码到具体操作的实现。

总共分为4个部分，simulator，memory，register和main.

其中main.cpp中总共列举了7中不同运行功能，包括help，file，register，single，begin，output和detail。

实验过程

（1）首先看memory部分，我们需要把文件所给的机器码准换成实际信息存入memory中。

其次要定义对memory进行存取操作的函数。

e.g.

```
namespace virtual_machine_nsp {
    void memory_tp::ReadMemoryFromFile(std::string filename, int beginning_address) {
        // Read from the file
        std::ifstream memoryfile;
        memoryfile.open(filename);
        int memory_ptr = beginning_address;
        int line_count = std::count(std::istreambuf_iterator<char>(memoryfile),std::istreambuf_iterator<char>(), '\n');
        memoryfile.close();
        memoryfile.open(filename);
        char temp[17];
        for (int i = 0; i < line_count; i++) {
            memoryfile >>temp;
            memory[memory_ptr] = 0;
            for (int j = 0; temp[j]; j++) {
                memory[memory_ptr]+=(temp[j]-48)* pow(2,16-j-1);
            }
            memory_ptr++;
        }
        memoryfile.close();
    }
    int16_t memory_tp::GetContent(int address) const {
        // get the content
        // TO BE DONE
        return memory[address];
    }
    int16_t& memory_tp::operator[](int address) {
        // get the content
        // TO BE DONE
        return memory[address];
    }
}; // virtual machine namespace
```

（2）simulator部分。

首先是位扩展函数，分为x为正数和x为负数两种情况。

其次是状态寄存器的变化，即nzp；引起变化的则是目的寄存器里写入的值的正负。

然后是对ADD，AND，NOT等各项指令执行过程的补充，

最后是虚拟机执行状态的变换，switch通过对指令opcode的选择，来完成不同指令和状态转换。

（3）main部分。

main.cpp中总共列举了7中不同运行功能，包括help，file，register，single，begin，output和detail。

（4）register部分比较简单，主要是编码了寄存器和状态，并且打印输出。