labA 实验报告

李远航
PB20000137

1. 实验内容

- 学习使用 makefile
- 阅读代码，理解代码的结构，锻炼阅读代码的能力
- 正确填写代码中`TO BE DONE`部分

2. 实验过程

(1) makefile 的使用

进入项目目录，输入`make`编译程序，`make clean`删除编译产生的相关工具链

(2) 代码补全过程

- 程序结构
  - 命令行参数
    - `-h` 显示帮助菜单
    - `-f` 输入文件路径
    - `-d` 输出 debug 信息
    - `-e` 输出错误信息
    - `-o` 输出文件路径
    - `-s` 十六进制模式
  - 程序运行过程
    - 第 0 遍扫描：将所有小写字母转换成大写，注释与代码分离
    - 第 1 遍扫描：找出所有的 label 并记录地址，找出程序的开始结束，记录每一行语句的地址
    - 第 2 遍扫描：根据每一行不同代码，进行翻译
- TO BE DONE
  - `trim`部分

```
inline std::string &RightTrim(std::string &s, const char *t = "
\t\n\r\f\v")
{
    int loc = s.length() - 1;
    int len = strlen(t);
    int flag = 0;
    while (1)
    {
        flag = 0;
        for (int i = 0; i < len; i++)
        {
            if (s[loc] == t[i])
            {
                loc--;
                flag = 1;
                break;
```

```
            }
        }
        if (flag == 0)
            break;
    }
    s = s.substr(0, loc + 1);
    return s;
}
```

- **int RecognizeNumberValue(std::string s)**

  将一个以#或者 X 开头的字符串转化成数字

```
int RecognizeNumberValue(std::string s)
{
    int ans = 0;
    int flag;
    if (s[0] == '#')
        flag = 10;
    else if (s[0] == 'X')
        flag = 16;
    if (s[1] != '-')
    {
        int l = s.length();
        for (int i = 1; i < l; i++)
            ans += pow(flag, l - 1 - i) * char2int(s[i]);
    }
    else
    {
        int l = s.length();
        for (int i = 2; i < l; i++)
            ans += pow(flag, l - 1 - i) * char2int(s[i]);
        ans = -ans;
    }
    return ans;
}
```

- **std::string NumberToAssemble(const int &number)**

  将数字转换成 16 位的二进制字符串

```
std::string NumberToAssemble(const int &number)
{
    std::string ans;
    for (int i = 15; i >= 0; i--)
        ans.push_back(((number & (1 << i)) >> i) + '0');
    return ans;
}
```

- **std::string NumberToAssemble(const std::string &number)**

  将字符串形式的数字转化成 16 位二进制字符串

```cpp
std::string NumberToAssemble(const std::string &number)
{
    int num = RecognizeNumberValue(number);
    std::string ans;
    for (int i = 15; i >= 0; i--)
        ans.push_back(((num & (1 << i)) >> i) + '0');
    return ans;
}
```

- **std::string ConvertBin2Hex(std::string bin)**

  将二进制字符串转化成 16 进制

```cpp
std::string ConvertBin2Hex(std::string bin)
{
    std::string result = "";
    for (int i = 0; i < 4; i++)
    {
        int num = 0;
        for (int j = 0; j < 4; j++)
        {
            num += (bin[4 * i + j] - '0') << (3 - j);
        }
        if (num <= 9 && num >= 0)
            result.push_back(num + '0');
        else
            result.push_back(num - 10 + 'a');
    }
    return result;
}
```

- **std::string assembler::TranslateOprand(int current_address, std::string str, int opcode_length)**

  根据参数返回字符串（寄存器或立即数）

```cpp
std::string assembler::TranslateOprand(int current_address,
std::string str, int opcode_length)
{
    str = Trim(str);
    auto item = label_map.GetValue(str);
    if (!(item.getType() == vAddress && item.getVal() == -1))
    {
        int offset_;
        if (item.getType() == vAddress)
        {
            offset_ = item.getVal() - current_address - 1;
            std::string ans;
            for (int i = opcode_length - 1; i >= 0; i--)
                ans.push_back(((offset_ & (1 << i)) >> i) + '0');
            return ans;
        }
    }
    if (str[0] == 'R')
    {
        int num_return = str[1] - '0';
        std::string ans;
```

```
        ans.push_back(((num_return & (1 << 2)) >> 2) + '0');
        ans.push_back(((num_return & (1 << 1)) >> 1) + '0');
        ans.push_back((num_return & 1) + '0');
        return ans;
    }
    else
    {
        int num_return = RecognizeNumberValue(str);
        std::string ans;
        for (int i = opcode_length - 1; i >= 0; i--)
            ans.push_back(((num_return & (1 << i)) >> i) + '0');
        return ans;
    }
}
```

- Convert `line` into upper case

  小写转换成大写

```
for (int i = 0; i < origin_line.length(); i++)
{
    if (line[i] <= 'z' && line[i] >= 'a')
        line[i] += 'A' - 'a';
}
```

- Split content and comment

  注释分离

```
std::string content_str = origin_line.substr(0, comment_position);
std::string comment_str = line.substr(comment_position + 1,
line.size() - comment_position - 1);
```

- 第 1 遍扫描时 `.FILL` 和 `.BLKW`

  `.FILL` 记录当前地址 `.BLKW` 将整体地址向后加

- 遇到指令行

  直接存储状态为指令，continue

- 遇到 label

  存储 label 和它的地址

- 遇到有 label 的 `.BLKW` 和 `.STRINGZ`

  记录 label，对地址进行偏移

- 翻译时遇到 `.BLKW`，识别占用的位数，输出

```
std::string num_blank;
line_stringstream >> num_blank;
int num_ = RecognizeNumberValue(num_blank);
std::string output_line = "0000000000000000";
if (gIsHexMode)
    output_line = ConvertBin2Hex(output_line);
output_file << output_line << std::endl;
for (int i = 1; i < num_; i++)
    output_file << output_line << std::endl;
```

- 翻译时遇到`.STRINGZ`，依次输出字符对应的 ascii 码的 16 位 2 进制字符串

```
std::string str;
line_stringstream >> str;
int len = str.length();
int num_ = str[0];
std::string ans_;
for (int i = 15; i >= 0; i--)
    ans_.push_back(((num_ & (1 << i)) >> i) + '0');
output_file << ans_ << std::endl;
for (int i = 1; i <= len; i++)
{
    int num = 0;
    if (i != len)
        num = str[i];
    std::string ans;
    for (int j = 15; j >= 0; j--)
        ans.push_back(((num & (1 << j)) >> j) + '0');
    output_file << ans << std::endl;
}
```

- 将逗号转化成空格

```
for (int i = 0; i < parameter_str.size(); i++)
    if (parameter_str[i] == ',')
        parameter_str[i] = ' ';
```

- 具体不同操作码的转化部分
  - **BR,BRN,BRZ,BRP,BRNZ,BRNP,BRZP,BRNZP** 基本相同

```
result_line += "0000000";
if (parameter_list_size != 1)
  return -30;
result_line += TranslateOprand(current_address,
parameter_list[0], 9);
break;
```

  - **JMP**

```
result_line += "1100000";
if (parameter_list_size != 1)
{
    // @ Error parameter numbers
    return -30;
}
result_line += TranslateOprand(current_address,
parameter_list[0]);
result_line += "000000";
break;
```

  - **JSR**

```
result_line += "01001";
if (parameter_list_size != 1)
{
    // @ Error parameter numbers
    return -30;
}
result_line += TranslateOprand(current_address,
parameter_list[0], 11);
break;
```

- **JSRR**

```
result_line += "0100000";
if (parameter_list_size != 1)
{
    // @ Error parameter numbers
    return -30;
}
result_line += TranslateOprand(current_address,
parameter_list[0]);
result_line += "000000";
break;
```

- 后续与前面基本相同，具体可见代码

3. 程序运行效果

以 DEBUG 模式运行程序可以在终端得到以下输出

```
Label Map:
Name: ARRAY [ Address -- 0x300f ]
Name: DONE [ Address -- 0x300d ]
Name: LOOP [ Address -- 0x3006 ]
Name: N [ Address -- 0x3023 ]
Name: OUTPUT [ Address -- 0x3024 ]

ffffffff .ORIG X3000
3000 AND R5, R5, #0
3001 LEA R0, ARRAY
3002 LD R1, N
3003 LDR R2, R0, #0
3004 NOT R2, R2
3005 ADD R2, R2, #1
3006 LOOP LDR R3, R0, #0
3007 ADD R3, R3, R2
3008 BRNP DONE
3009 ADD R0, R0, #1
300a ADD R1, R1, #-1
300b BRP LOOP
300c ADD R5,  R5, #1
300d DONE ST R5, OUTPUT
300e HALT
300f ARRAY .BLKW #20
3023 N .FILL #20
3024 OUTPUT .BLKW #1
ffffffff .END
```

4. 实验收获

- 学习了 makefile 的使用
- 熟悉了汇编码转机器码的过程
- 锻炼了阅读代码的能力
- 对 LC-3 语言有了进一步的认识