计算机系统原理实验报告

MIPS 汇编

王子腾 3180102173

一、 实验描述

用 C 或 JAVA 编写程序，将 MIPS 汇编指令转换为机器码，并以二进制方式存盘，以作为随后反汇编的输入及 CPU 模拟器的执行程序。

实际汇编指令到机器码的转换。可以选择多种输入输出方式，可以是单条指令即输即显。也可以是选写好汇编源程序文件，以文件读入二进制文件输出。说明文档：所实现的指令、程序框图、使用方法、实例分析等等。

- 采用尽可能多的指令。
- 考虑伪指令。
- 考虑符号[标号、变量]
- 考虑表达式[如：SW $s1, 4*12($s0)]

二、 程序实现的指令

本程序实现的指令如下（所有指令支持大小写）

- **R 类型指令**

  add rd, rs, rt

  sub rd, rs, rt

  slt rd, rs, rt

  or rd, rs, rt

- **I 类型指令**

  lw rt, expr(rs)

  sw rt, expr(rs)

  addi rt, rs, expr

  beq rs, rt, label

  bne rs, rt, label

- **J 类型指令**

  j label

  jr rs

- **伪指令**

  move r1, r2

blt r1, r2, label

bgt r1, r2, label

ble r1, r2, label

bge r1, r2, label

三、 程序设计思路

- **输入与输出**

在本实验中，用户应将指令存储于文件夹目录下“assembly.txt”文件中，每条指令占一行，允许存在空行。在执行编译完毕后，程序将二进制指令输出并保存到同一目录下的“machine.txt”中。

使用 C 语言标准库中的 fgets()函数处理文件中每一行指令，并输入到程序中，对于空行，识别'\n'并跳过主干部分，继续读入，直至触碰到 EOF 后结束程序。

- **标签处理**

通过复制每段指令的前四位操作符，并作为输入，传入 classification()函数中进行分类，通过比对，将读入的字符串转入到不同类别的处理函数中进一步分解。

- **指令处理**

对于不同类别的指令，分为 R()（寄存器类型指令），I()（立即数类型），J()（转移类型），P()（伪指令）四个函数分别处理。对于寄存器对应数值的获取，通过 Reg()函数按所需位置返回获得的寄存器值，并通过 InReg()函数将寄存器值存入机器语言中；对于立即数的读取，通过 atoi()函数先转换为整数，再通过 itoa()函数转换成二进制格式字符串后，与输出字符串指定位置进行连接。

R 类型：

op	rs	rt	rd	shamt	func
6	5	5	5	5	6

当字符串分流至 R()函数时，先调用 Reg()，InReg()函数组合进行三次寄存器处理，将其同‘op‘插入到 mac[ ]字符串中，继而通过 switch 分出’add’, ‘sub’, ‘slt’, ‘or’四种特定操作后，再填入之后的位数。

I 类型：

op	rs	rt	Immediate
6	5	5	16

对‘op’, ‘rs’, ‘rt’的处理与 R 类型指令相似，而对于 immediate 处的数字或表达式，先检测有无运算符号，对运算结果处理完后在进行数字的转换类型和禁止操

作，具体使用的函数为 atoi()以及 itoa()函数，最后将这一部分截取 16 位后，存放至输出字符串中。

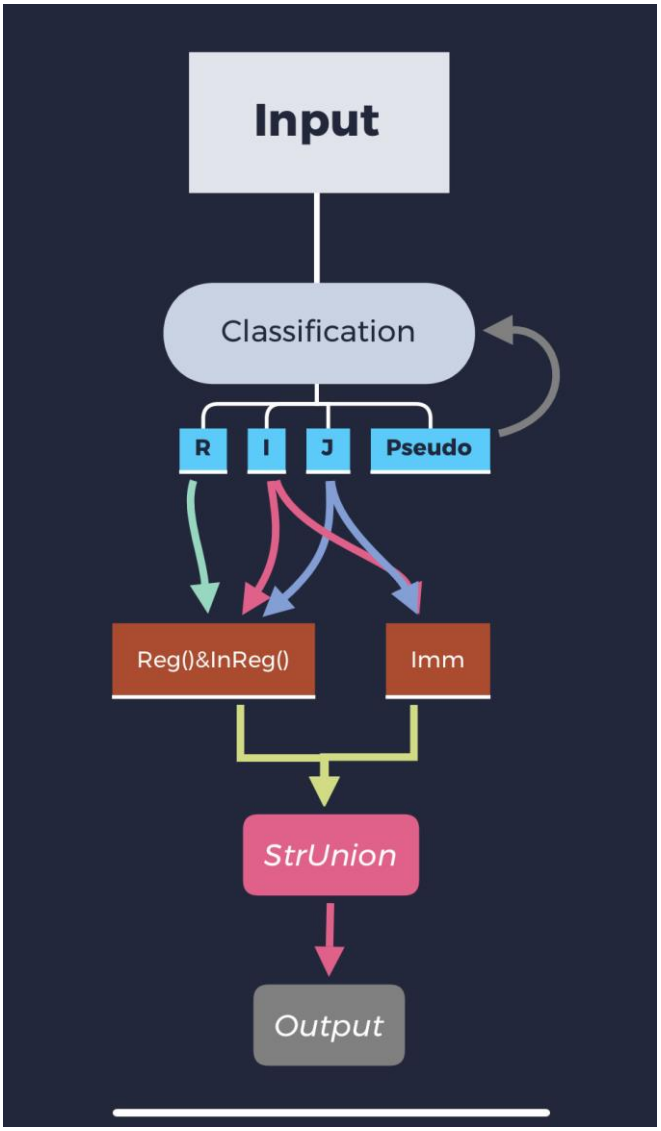
J 类型：

本程序实现的 J 函数有 j 和 jr 两种，处理时只需对两者分开操作即可，其中 j 的数字处理与 I 类中立即数相似，而 jr 则与 R 中的寄存器处理类似。

P 类型（伪指令）：

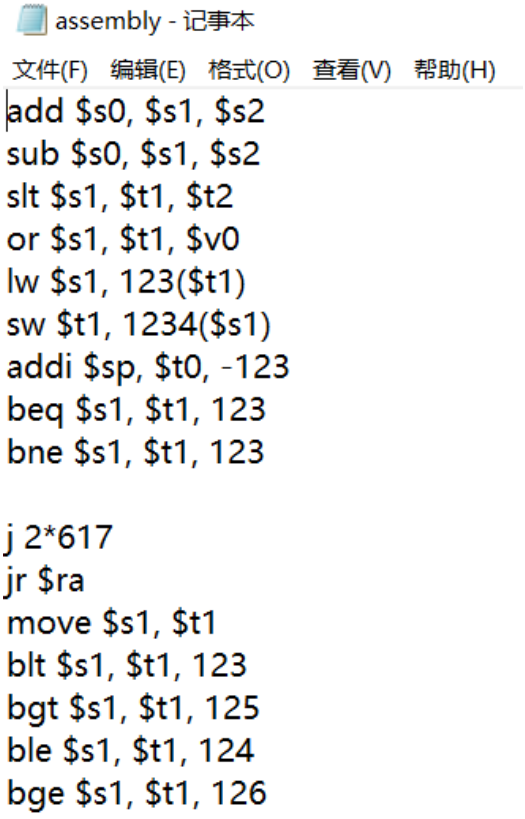
伪指令的实现主要是通过将 ins 字符串中存储的指令分别转换为替代指令，并通过多次循环 classification()函数实现一条伪指令转换为多条普通指令执行的过程，在字符串两次调用 classification()函数过程中，需要注意保留伪指令中传入的有效信息，并更新负责指令存储的 ins[]字符串。

四、 程序框图

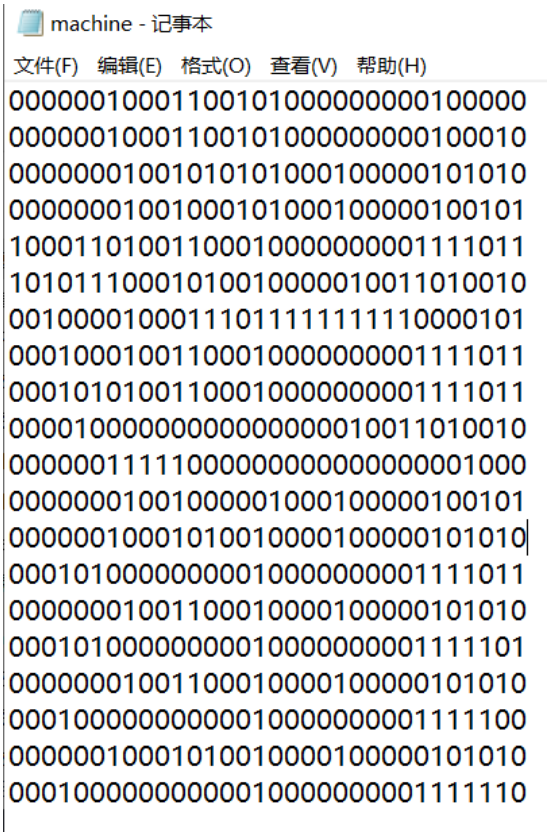


五、 实例分析

在汇编文件“assembly.txt“中输入如下指令：



得到在‘machine.txt’中的二进制代码为：



经比对，二进制代码正确，实验与预期符合。

六、 实验心得

在本次实验中，由于 C 语言字符串处理灵活度较低，因此在处理过程中花费了一些时间尝试，最后选取 strcat()和 strncpy()进行有效拼接和赋值，同时，对指令的分类和标签的处理是本次实验比较重要的部分，我选取了 C 语言中的枚举 enum{}来存储相应标签，使得程序的可读性大大提高，为后期检查错误节省了时间。通过本次实验，我对 R 类型、I 类型、J 类型以及伪指令有了更深的理解，对寄存器的用途也更加熟悉。