实验报告：中间代码到MIPS汇编的转换器

1. 程序功能概述

本程序实现了一个将中间代码转换为MIPS汇编代码的编译器后端模块，主要功能包括：

1. **寄存器分配**：实现了基于简单溢出策略的寄存器分配算法，能够处理寄存器不足时的变量溢出到内存的情况。
2. **指令翻译**：支持多种中间代码指令到MIPS汇编的转换，包括：
   - 算术运算（ADD, SUB, MUL, DIV）
   - 赋值操作（ASSIGN, ADDR, DEREF_L, DEREF_R）
   - 控制流（LABEL, GOTO, IF_GOTO）
   - 函数调用（FUNCTION, CALL, RETURN, PARAM, ARG）
   - 输入输出（READ, WRITE）
   - 内存分配（DEC）
3. **函数调用处理**：正确处理函数调用时的参数传递、寄存器保存与恢复、返回值处理等。
4. **变量管理**：维护变量描述符链表，跟踪变量在寄存器或内存中的位置。
5. **内存管理**：处理变量溢出到内存的情况，生成相应的加载/存储指令。

2. 实现亮点

2.1 寄存器分配策略

实现了基于优先级的寄存器分配算法，分配策略如下：

1. 首先尝试分配空闲寄存器
2. 优先溢出常量值
3. 其次溢出临时变量
4. 最后溢出普通变量
5. 采用最近最少使用策略选择要溢出的寄存器

```
int allocate(pOperand op, FILE *fp) {
    // 查找空闲寄存器
    for (int i = 8; i < 26; i++) {
        if (!_reg[i].state) {
            _reg[i].state = 1;
            add_var_desc(i, op, fp);
            return i;
        }
    }
    // 溢出策略实现...
}
```

2.2 变量溢出处理

当寄存器不足时，能够将变量溢出到内存，并维护溢出变量的信息：

```
void spill_to_memory(FILE *fp, struct VarDesc *var_desc) {
    // 生成唯一标签
    char *label = malloc(32);
    sprintf(label, "_spill_%d", spill_counter++);

    // 在数据段声明变量
    fprintf(tmpout, "%s: .word 0\n", label);

    // 存储到全局地址
    fprintf(fp, "  sw %s, %s\n", _reg[var_desc->reg_num].name, label);

    add_spill_mem_var_desc(var_desc->op, label);
}
```

2.3 函数调用处理

正确处理函数调用时的参数传递和寄存器保存：

```
case CALL: {
    // 保存返回地址和寄存器
    fprintf(fp, "  addi $sp, $sp, -4\n");
    fprintf(fp, "  sw $ra, 0($sp)\n");
    store_reg(fp);

    int param_reg_num = 0;
    while (tmp != NULL && tmp->kind == ARG) {
        if (param_reg_num < 4) {
            int reg_num = ensure(tmp->u.singleop.op, fp);
            fprintf(fp, "  move %s, %s\n",
                   _reg[4 + param_reg_num].name, _reg[reg_num].name);
        } else {
            fprintf(fp, "  addi $sp, $sp, -4\n");
            fprintf(fp, "  sw %s, 0($sp)\n", _reg[4 + param_reg_num].name);
        }
        param_reg_num++;
    }

    fprintf(fp, "  jal %s\n", curr->u.call.op->u.name);
    // ...恢复寄存器等操作
    break;
}
```

3. 编译与使用方法

3.1 编译方法

程序可以通过Makefile进行编译

在lab4目录下执行以下命令：

```
make

./parser ../Test/Test1 out.s
```