编译原理Lab4目标代码生成 实验报告

161220188 朱寅 161220182 周宇航

实验任务

将中间代码翻译成MIPS32代码（机器指令级代码）

实验运行

使用文件夹下的parser文件，在命令行终端下使用指令

./parser test.cmm out.s

实验过程

给中间代码的变量ti和vi都分配在内存栈中的空间需要的位置，包括距离$fp的偏移量，存值的寄存器等

```
typedef struct Variable_* Variable;
typedef struct Variable_ {
    char *varName;
    int offset;
    int regId;
    struct Variable_* next;
} Variable_;
```

寄存器的结构

```
typedef struct Register {
    char* regName;
    int used;
    Variable content;
} Register;
```

本次实验只需要对中间代码进行遍历以及分析翻译即可

参数传递

我们的参数传递使用的是首选a0~a3在加上开辟栈空间的方式存放args

```
InterCodes printArgCode(InterCodes current) {
    InterCodes c = current;
    for(; c->code.kind == ARG; c = c->next)
        argCount += 1;
    int minus = 0;
    if(argCount - 5 > 0) {
        minus = argCount - 5;
```

```
        printf("subu $sp, $sp, %d\n", minus*4);      //这里得出参数个数，决定是否开辟栈空间
    }
    int i = argCount;
    for(InterCodes c1 = c->prev; c1->code.kind == ARG; c1 = c1->prev)
    {
        ......
        Variable aim = searchVariable(name);
        if(aim == NULL) {
            exit(-1);
        }
        else {                                  //按照顺序取用a类寄存器和内存位置
            if(i == argCount)
                printf("lw $a0, %d($fp)\n", aim->offset);
            else if(i == argCount - 1)
                printf("lw $a1, %d($fp)\n", aim->offset);
            else if(i == argCount - 2)
                printf("lw $a2, %d($fp)\n", aim->offset);
            else if(i == argCount - 3)
                printf("lw $a3, %d($fp)\n", aim->offset);
            else
            {
                printf("lw $t0, %d($fp)\n", aim->offset);
                printf("sw $t0, %d($sp)\n", 4*(argCount-4-i));
            }
        }
        i = i - 1;
    }
    return c->prev;
}
```