Decaf报告

Decaf的工作包括了两个部分：
第一个是生成RiscV的汇编代码，这个也是编译原理课程的要求。
第二个是用高级语言（C语言）实现Decaf的内置函数。

Decaf 后端代码生成

在Decaf框架中，后端代码生成的流程是：
1.对之前生成的中间代码（TAC）做数据流分析；
2.接着完成寄存器分配，将TAC使用的虚拟寄存器转变成物理寄存器；
3.使用目标架构的指令（RiscV指令集）完成TAC需要的操作。

生成汇编代码

为了使得Decaf框架生成的是RiscV的汇编代码，在原有的框架基础上做了一些改动。

基本工作是将代码中Mips指令全都改成RiscV的指令。
其中包括一些细节，比如修改寄存器名，以及将一些RiscV里没有的指令拆解成RiscV中的指令。

运行时

以上过程生成的汇编代码并不能运行，因为这个汇编代码会使用Decaf自带的内置函数，而这些内置函数还没有被实现。

因此，我们使用 C 语言实现这套内置函数，生成了一个函数库（o文件），并将此用于与Decaf生成的汇编代码的链接。
预定计划如此，但实际实现中链接方式出了差错，所以流程便卡在了生成的RiscV汇编代码上。

生成的汇编代码：

```
# intrinsic library
_PrintInt:
          lw    a0, 4(sp)
          tail  _wrjlibc__PrintInt
          jr    ra
_PrintString:
          lw    a0, 4(sp)
          tail  _wrjlibc__PrintString
          jr    ra
_PrintBool:
          lw    a0, 4(sp)
          tail  _wrjlibc__PrintBool
          jr    ra
_Alloc:
          lw    a0, 4(sp)
          tail  _wrjlibc__Alloc
          jr    ra
_Halt:
          tail  _wrjlibc__Halt
          jr    ra
_StringEqual:
          lw    a0, 4(sp)
          lw    a1, 8(sp)
          tail  _wrjlibc__StringEqual
          jr    ra
```

可以看到汇编代码使用了内置函数

```
main:                                   # function entry
# prolog
          sw s0, 0(sp)
          sw ra, -4(sp)
          move s0, sp
          addi sp, sp, -36
# end prolog
_L21:
          la    s2, _STRING4
          sw    s2, 4(sp)
          call  _PrintString
          call  _Cat_New
          move  s2, a0
          mv    s3, s2
          la    s2, _STRING5
          sw    s2, 4(sp)
          sw    s3, -8(s0)
          call  _PrintString
          lw    s3, -8(s0)
          la    s2, _Cat
          lw    s4, 0(s3)
          sw    s2, -12(s0)
          sw    s4, -16(s0)
          sw    s3, -8(s0)
_L22:
          lw    s2, -12(s0)
          lw    s3, -16(s0)
          sub   s4, s2, s3
          seqz  s4, s4
          sw    s2, -12(s0)
          sw    s3, -16(s0)
          bne   s4, zero, _L25
```

主函数本体

使用C语言实现的函数库：

```
void __PrintInt(int v) {
    int len = 10;
    int neg = 0;
    if (v < 0) {
        neg = 1;
        v = -v;
    }
    do {
        buf[len--] = v % 10 + '0';
        v /= 10;
    } while (v > 0);
    if (neg)
        buf[len--] = '-';
    write_flushed(1, buf + len + 1, 10 - len);
}

void __PrintString(char* v) {
    int len = 0;
    while (v[len] != 0) len++;
    write_flushed(1, v, len);
}

const char* truestr = "true";
const char* falsestr = "false";
void __PrintBool(int v) {
    if (v) {
        write_flushed(1, truestr, 4);
    } else {
        write_flushed(1, falsestr, 5);
    }
}
```