# 1.添加语句

【问题描述】

（1）小译同学在编写testfile时希望能写出 int i = getint(); 这类输入语句。

（2）课程组提供的 SysY 文法中没有位运算操作，现在需实现一个“按位与”的运算操作，并用关键字 bitand 来表示该操作，如表达式 b bitand c 表示 变量 b 与 变量 c 的数值做“按位与”运算得到的结果。

 本次考核中，要求同学们在代码生成作业的基础上实现上述两项需求。

【题目要求】

（1）增加的语法成分如下所示：

    (i) 变量定义 VarDef → Ident { '[' ConstExp ']' }

                        | Ident { '[' ConstExp ']' } '=' InitVal

                        | Ident '=' 'getint' '(' ')'

    (ii) 乘除模表达式 MulExp → UnaryExp | MulExp ('\*' | '/' | '%' | 'bitand' ) UnaryExp

（2）为了降低难度，使用输入语句进行变量定义时，变量**只可能**为一个普通 int 类型的变量，**不会**出现对数组变量进行赋值，如 int arr[2] = getint(); 这是不合法的

（3）为了降低难度，按位与运算符号 & 被关键字 bitand 代替，运算效果与 C/Java 语言程序相同，但运算优先级与**乘除模**('\*','/','%')为同一优先级，与 C/Java 语言程序规定的位运算优先级**不同**，如 a + b bitand c \* d; 按照运算优先级翻译成中间代码为

      t1 = b bitand c

      t2 = t1 \* d

      t3 = a + t2

（4）为了降低难度，常量表达式(ConstExp)的计算中**不会**出现按位与(bitand)操作，如 const int p = N bitand M; 和 int a[N bitand M]; （其中 M 和 N 为常量），这些都是不合法的

（5）在新增的语法规则中，bitand 为保留关键字，测试样例中**不会**出现标识符 Ident 为 bitand 的情况

（6）int i = getint(); 等价于 int i; 与 i = getint(); 两条语句

（7）a bitand b 的运算效果等价于 C/Java 语言程序中的 a & b

（8）提示：按位与运算可选用 `and` 指令，其格式与 `add`、`sub`、`mul` 等指令相同。

【输入形式】testfile.txt为符合文法要求的测试程序。另外可能存在来自于标准输入的数据。

【输出形式】按照选择的不同目标码分为三类：

  1）生成MIPS的编译器

     按如上要求将目标代码生成结果输出至mips.txt中。

  2）生成LLVM IR的编译器

     按如上要求将目标代码生成结果输出至llvm\_ir.txt中。

  3）生成PCODE的编译器

     按如上要求生成PCODE并解释执行，在pcoderesult.txt中记录解释执行结果。

【测试样例】

int main()

{

int i = getint(), j = getint();

printf("%d", i bitand j);

return 0;

}

【样例输入】

5

9

【样例输出】

1

【样例说明】

// i = 5(00000101), j = 9(00001001)

// 按位与结果为 1(00000001)

【评分标准】按与预期结果不一致的行数扣分，每项扣10%。

【目标代码说明】

 （1）PCODE代码的定义可参见教材P458, Pascal-S指令代码集，可以进行修改，解释执行程序也可以借鉴Pascal-S编译器源码中的解释执行程序，若PCODE代码修改了请相应修改解释执行程序

 （2）MIPS代码可以选择基础指令及伪指令，不能选择宏指令； MARS 使用 4.5 版本（课程组修改版本），请在教学平台的“课件下载”中获取课程组修改过的版本Mars-for-Compiler-2022.jar以及“竞速排序及仿真器使用说明2022”文档查看具体要求

【测试样例说明】

   本次考核共有15个测试点，均 Accept 即可拿到满分。

| **测试点** | **分级** | **说明** |
| --- | --- | --- |
| testfile1 | C级样例 | 不涉及新增文法 |
| testfile2 |
| testfile3 |
| testfile4 | 仅增加形如int i = getint()内容 |
| testfile5 |
| testfile6 | 仅增加bitand内容 |
| testfile7 |
| testfile8 | B级样例 | 不涉及新增文法 |
| testfile9 | 仅增加形如int i = getint()内容 |
| testfile10 | 仅增加bitand内容 |
| testfile11 | 增加全部两项内容 |
| testfile12 | A级样例 | 不涉及新增文法 |
| testfile13 | 仅增加形如int i = getint()内容 |
| testfile14 | 仅增加bitand内容 |
| testfile15 | 增加全部两项内容 |

请编译器生成的目标代码为PCODE或MIPS汇编的同学完成本题。

# 2.强测

请编译器生成的目标代码为PCODE或MIPS汇编的同学完成本题。

【作业描述】

  本题目针对平时作业的文法，用如下两个测试程序testfile1-testfile2，对生成PCODE与LLVM IR的同学考核代码生成，对生成MIPS的同学考核代码生成及代码优化，并在此基础上回答简答题。

【输入形式】testfile.txt为符合文法要求的测试程序，另外可能存在来自于标准输入的数据。

【输出形式】按照选择的不同目标码分为三类：

  1）生成PCODE的编译器

    生成PCODE并解释执行，在pcoderesult.txt中记录解释执行结果。

  2）生成LLVM IR的编译器

    将目标代码生成结果输出至llvm\_ir.txt中。

  3）生成MIPS的编译器

    a) 将目标代码生成结果输出至mips.txt中。

   b) 将第一个测试程序优化前后的中间代码分别输出到命名为“testfile1\_学号\_姓名\_优化前/后中间代码.txt”的文件中，优化前后的目标代码分别输出到命名为“testfile1\_学号\_姓名\_优化前/后目标代码.txt”的文件中，并在下面的文件上传题中提交，提交到本题目中的编译器需为产生优化后目标代码的版本。如果第一个测试程序没有通过，则提交第二个测试程序优化前后的中间代码与目标代码，对应文件命名也改为 testfile2。

【评分要求】代码生成部分：按与预期结果不一致的行数扣分，每项扣10%。

        代码优化部分：根据FinalCycle = DIV \* 50 + MULT \* 4 + JUMP/BRANCH \* 1.2 + MEM \* 2  + OTHER \* 1的值排名，FinalCycle越小排名越靠前。根据各测试程序的排名得分（不公布排行榜）。

【testfile1】

int a,b,c,d,e,i=0;

int func0(int a,int b) {

    int i = 0;

    int c = 128;

    return (a+b)%c;

}

int func1(int a) {

    i = i + 1;

    return i;

}

int func2(int a,int b) {

    if (a % b == 0) {

        return 1;

    }

    return 0;

}

int func3() {

    printf("glo\_i = %d\n",i);

    int tt0,tt1,t2,tt3,tt4,v=1906;

    while (i < 10000) {

        int v = a \* 4 \* 32 \* a / a / 32;

        b = func0(b,v);

        tt0 = a\*4 + b + c ;

        tt1 = a\*4 + b + c + d;

        t2 = a\*4 + b + c + d + e;

        tt3 = a\*4 + b + c + d + e;

        tt4 = a\*4 + b + c + d + e;

        if (func2(i,1000)) {

            printf("sum = %d\n", tt0 + tt1 + t2 + tt3 + tt4);

        }

        func1(i);

    }

    return tt0 + tt1 + t2 + tt3 + tt4;

}

int main() {

    int i = 0;

    a = getint();

    b = getint();

    c = getint();

    d = getint();

    e = getint();

    i = getint();

    printf("main\_i = %d\n",i);

    printf("%d\n",func3());

    return 0;

}

【testfile2】

int interesting = 1;

const int MOD = 100005;

const int SUM = 777777;

int one(){

    return 1;

}

int one2(int i){

    if(i > 2\*i+1){

        return i;

    }

    return one() + one()/one() - one();

}

int tRue(){

    int n = -99;

    while(1==1){

        if(n < one()){

            n = n + 1;

        }

        else {

            return n;

        }

    }

    while(1==1){

        if(n < one2(one())){

            n = n + 1;

        }

        else {

            break;

        }

    }

    while(1==1){

        if(n+one2(n) < one2(one2(one()+1))){

            n = n + 1;

        }

        else {

            return n;

        }

    }

    return n;

}

int fAlse(){

int x = tRue();

while(!0){

x = x + tRue()\*tRue();

printf("%d\n",x);

if(x > SUM%123){

break;

}

}

return x % tRue();

}

int main(){

    int x;

    x = getint();

    while(x%MOD < tRue()+fAlse()){

        printf("%d\n",tRue());

        x = x + 100;

        while(x/MOD+MOD < tRue()+fAlse()){

            x = x - 1;

            printf("%d\n",tRue()+x);

            while(x/MOD < tRue()+fAlse()){

                x = x - 1;

                printf("%d\n",tRue()+x);

            }

        }

    }

    printf("%d\n",x);

    return 0;

}

# 3.错误处理

请编译器生成的目标代码为PCODE或MIPS汇编的同学完成本题。

所有同学均需完成本题。

在对**一维数组**赋初值时，如果用于赋初值的元素个数与该数组声明的元素个数不一致（大于或大于），如 int a[2] = {0}; int b[1] = {0,1};是不合法的。请结合你的编译器**源代码**进行说明，你可以进行怎样的处理来检测这类新的错误（**不需要**考虑二维数组赋初值情况）。

# 4.简答题（非MIPS）

请编译器生成的目标代码为PCODE或LLVM IR的同学完成本题。

题目：请详细说明你的编译器为代码生成及代码优化题目中以下测试程序的函数设计的运行栈（活动记录）：

|  |  |
| --- | --- |
| 测试程序 | 函数名 |
| testfile1.txt | int func3() |

请编译器生成的目标代码为PCODE或LLVM IR的同学完成本题。

# 4．简答题（MIPS）

请编译器生成的目标代码为MIPS汇编的同学完成本题。

题目：请详细说明你的编译器给代码生成及代码优化题目中以下测试程序的函数设计的寄存器分配策略，包括全局寄存器和临时寄存器。

|  |  |
| --- | --- |
| 测试程序 | 函数名 |
| testfile1.txt | int func3() |

请编译器生成的目标代码为MIPS汇编的同学完成本题。

# 5.MIPS代码

请编译器生成的目标代码为MIPS汇编的同学完成本题。

请为本次考核中公开的第一个测试程序生成优化前/后的中间代码和优化前/后的目标代码（若第一个没通过，则生成第二个测试程序的优化前后代码），并将所有文件压缩为命名为"学号.zip”的文件，提交该zip文件。

优化前后的中间代码命名为：testfilei\_学号\_姓名\_优化前/后中间代码.txt

优化前后的目标代码命名为：testfilei\_学号\_姓名\_优化前/后目标代码.txt

若没有实现优化，也需要提交此次作业，只提交优化前的代码即可。

请编译器生成的目标代码为MIPS汇编的同学完成本题。

