ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences
智能软件研究中心
Intelligent Software Research Center

面向RISC-V的Tiger编译器后端实现

程序语言与编译技术实验室 韩柳彤

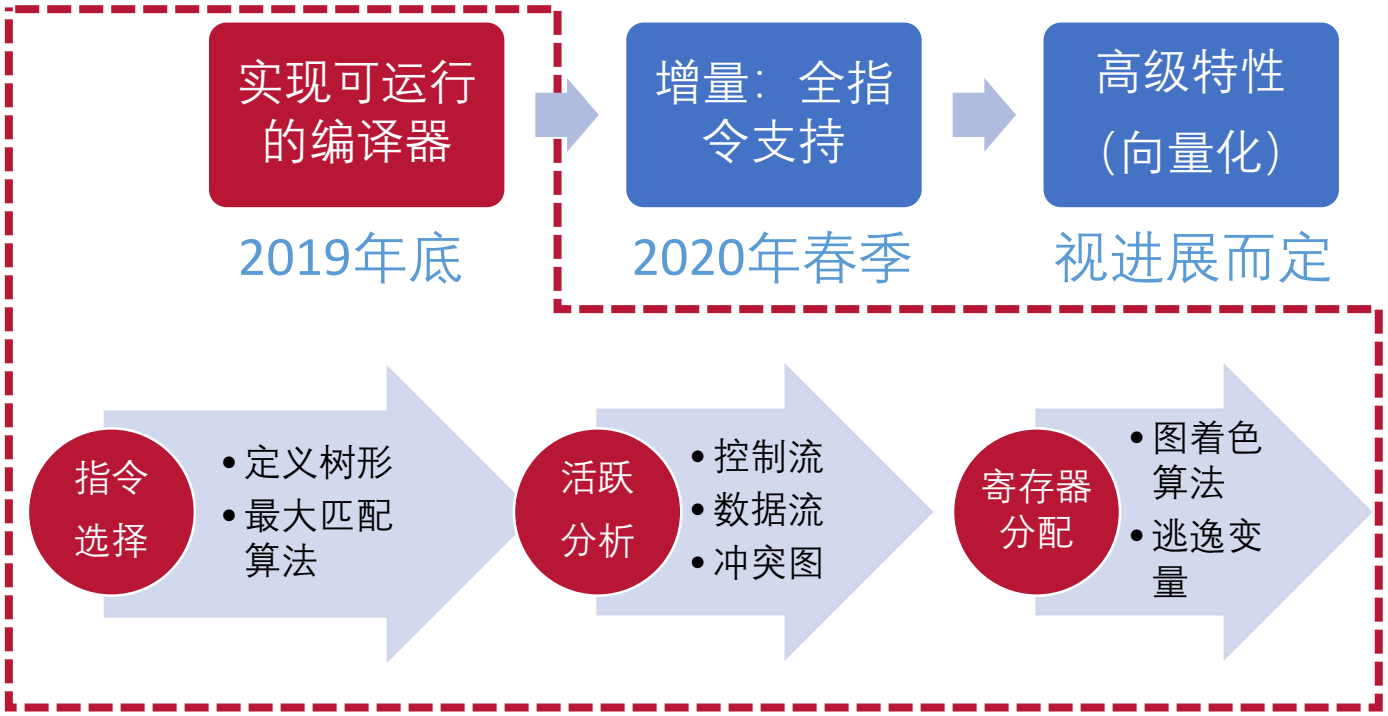
2019/12/18

面向RISC-V的Tiger编译器

- 目标：提供一个可运行的教学用编译器实现方案
- Tiger语言
  - 《现代编译原理》中定义的小规模语言Tiger（教学用）
  - 嵌套函数、数组、整型、字符串以及几种简单的控制结构
- RISC-V指令集架构
  - 开源
  - 模块化设计
  - 简洁 (没有向前兼容的历史包袱)

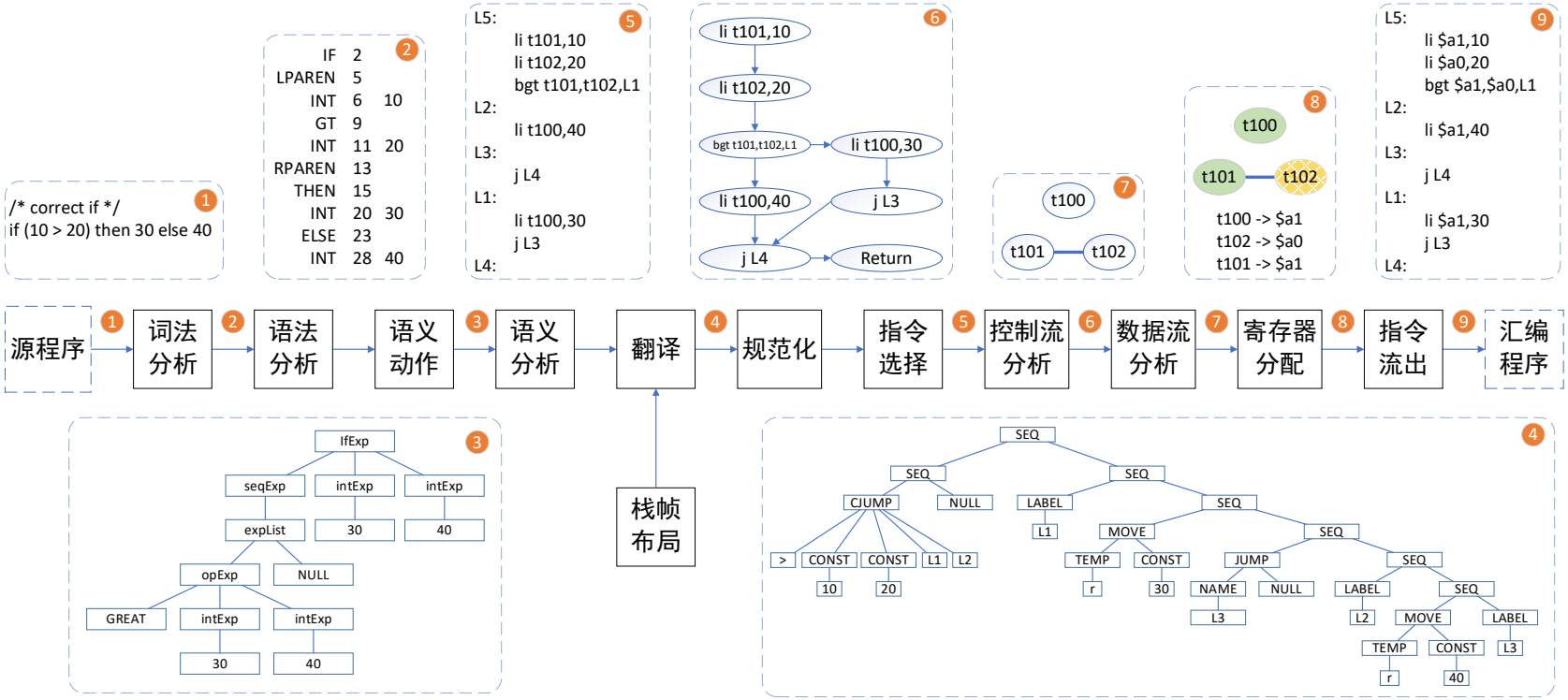
ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences
智能软件研究中心 Intelligent Software Research Center

增量开发模式:

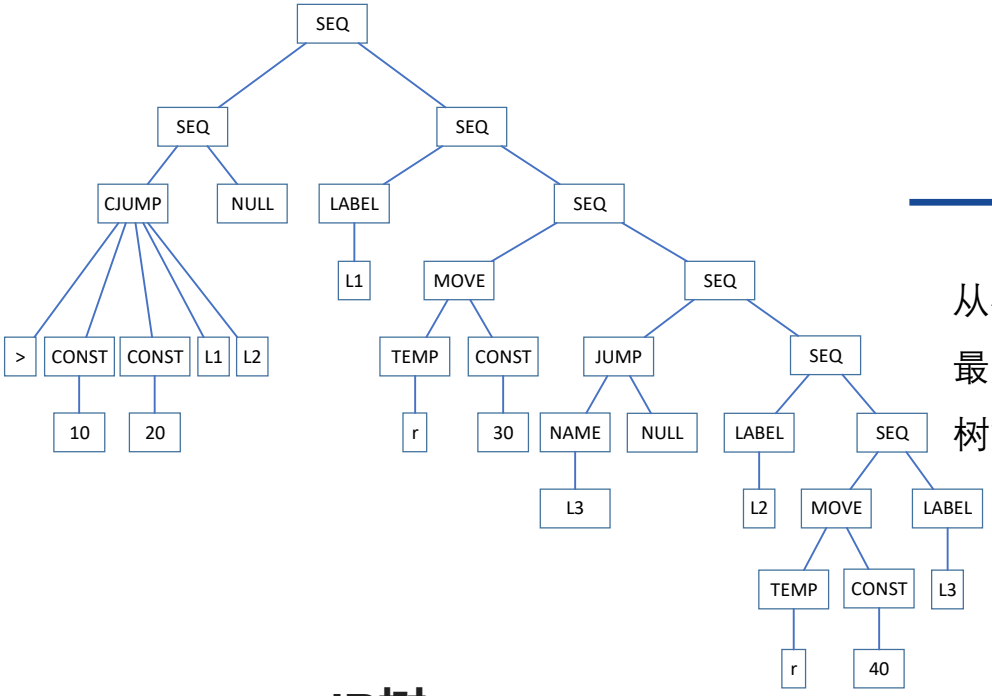


02 Tiger编译器架构介绍

编译器各模块及阶段示例



指令选择



IR树

最大匹配算法

从树根节点开始，选择适合的最大树形覆盖节点，对余下子树重复执行该算法。

```
li t101,10
li t102,20
bgt t101,t102,L1
L2:
li t100,40
L3:
j L4
L1:
li t100,30
j L3
L4:
```

含有临时变量的汇编指令代码

活跃分析

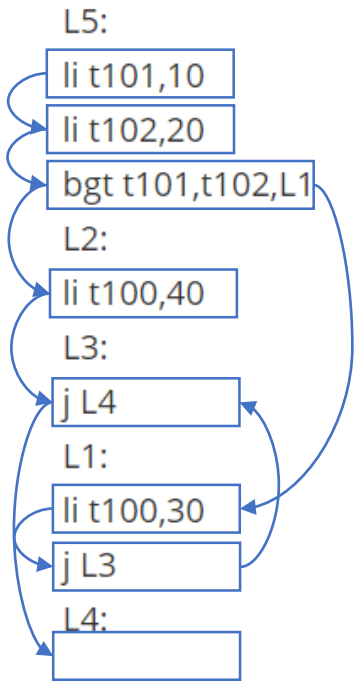
图中的每个节点代表一条指令，如果指令n的执行可以跟随在指令m之后，则图中会有一条边（m , n）。

构造程序的控制流图，用于进行数据流分析，得到活跃区间。

```
L5:
li t101,10
li t102,20
bgt t101,t102,L1
L2:
li t100,40
L3:
j L4
L1:
li t100,30
j L3
L4:
```

含有临时变量的汇编指令代码

构造控制流图



控制流图

活跃分析

(0): 1
(1): 2
(2): 5 3
(3): 4
(4): 7
(5): 6
(6): 4
(7):

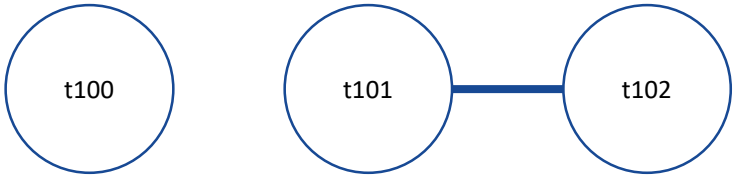
控制流图
(以邻接表形式存储)

$$in[n] = use[n] \cup (out[n] - def[n])$$

$$out[n] = \bigcup_{s \in succ} in[s]$$

解数据流方程

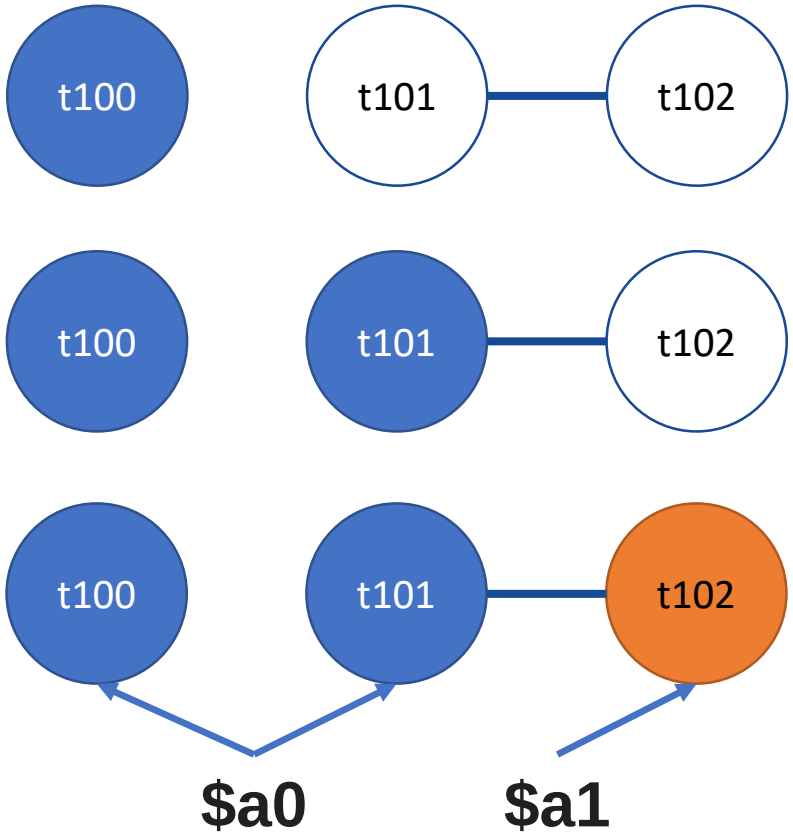
	0		1		2		3		4		5		6	
	in	out	in	out	in	out	in	out	in	out	in	out	in	out
0	\	\	\	\	\	\	\	\	\	t101	\	t101	\	t101
1	\	\	\	\	\	t101 t102	t101	t101 t102	t101	t101 t102	t101	t101 t102	t101	t101 t102
2	\	\	t101 t102	\	t101 t102	\	t101 t102	\	t101 t102	\	t101 t102	\	t101 t102	\
3	\	\	\	\	\	\	\	\	\	t100	\	t100	\	t100
4	\	\	\	\	\	t100	t100	t100	t100	t100	t100	t100	t100	t100
5	\	\	\	\	\	\	\	\	\	\	\	t100	\	t100
6	\	\	\	\	\	\	\	t100	t100	t100	t100	t100	t100	t100
7	\	\	t100	\	t100	\	t100	\	t100	\	t100	\	t100	\



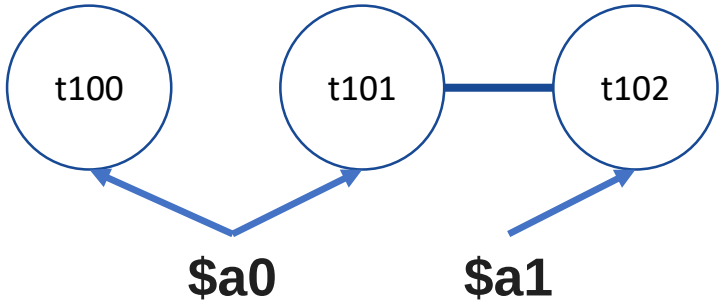
冲突图

寄存器分配

使用图的着色算法进行寄存器分配，给冲突图进行着色。假设目标机器有k个寄存器，只要冲突图是k可着色的，就可以将寄存器合理的分配给不同的临时变量，否则，需要溢出图中某个变量到内存里，直到新的冲突图是k可着色的。



代码流出




```
L5:
li t101,10
li t102,20
bgt t101,t102,L1
L2:
li t100,40
L3:
j L4
L1:
li t100,30
j L3
L4:
```

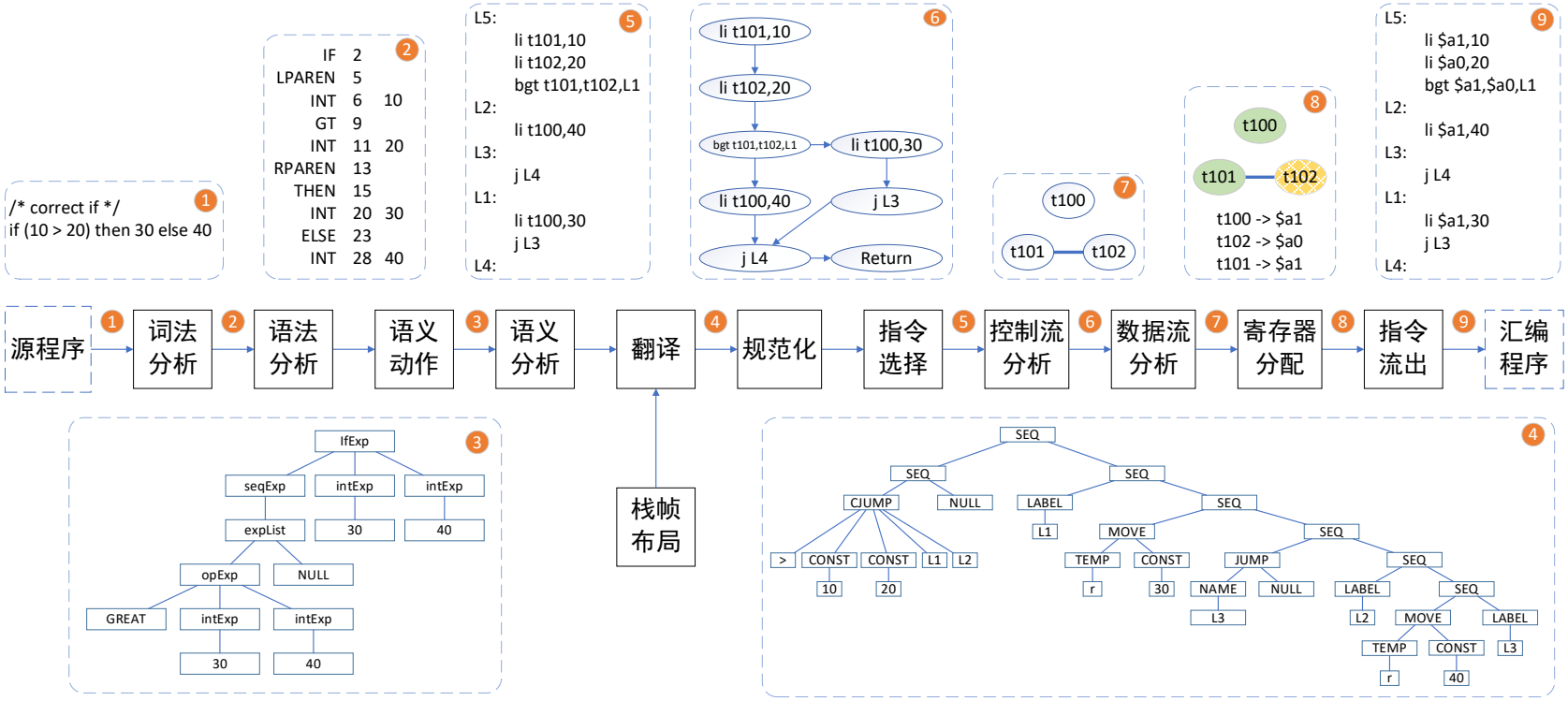
+

```
t100 -> $a1
t102 -> $a0
t101 -> $a1
```

=

```
L5:
li $a1,10
li $a0,20
bgt $a1,$a0,L1
L2:
li $a1,40
L3:
j L4
L1:
li $a1,30
j L3
L4:
```

编译器各模块及阶段示例



项目展望

- 完善基础部分
- 实现编译器的高级功能
  - 实现垃圾回收等高级功能，作为进阶内容为分层教学提供支持。
- 开发在线实训平台
  - 提供代码框架、运行环境和自动代码测试、赋分反馈功能，帮助提升实践课程效果。

- 整体

- [ ] 使用现有测试用例，在 tiger/tiger-compiler库中部署CI
  - 期望的效果是每次Commit/MR之后，进行自动化测试，并输出测试结果
  - tiger源码提供了49个单元测试和2个系统测试
  - 可以对编译教学平台提供支持/积累经验
- [ ] 使用lint，检查代码规范并修正

- 前端

- [ ] 解决词法分析模块中字符串注释的匹配问题
- [ ] 解决语法分析模块中的移进/归约冲突
- [ ] 实现错误恢复功能 （虎书p54 3.5），完成Error Message模块
- [ ] 补全翻译成IR模块

- 后端

- [ ] 实现对函数调用的支持
- [ ] 补全指令选择模块
- [ ] 使用动态规划算法替换最大匹配算法（指令选择模块）
- [ ] 优化寄存器分配模块，实现溢出

谢 谢

2019/12/18

ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences
智能软件研究中心
Intelligent Software Research Center

Obfuscator-LLVM 编译混淆简介

程序语言与编译技术实验室 韩柳彤

2019/12/18

01 软件保护技术简介

目标：保护静态或运行中的代码的完整性和机密性。

两个方向：

- 加密：使用密钥加密方式　　阻止逆向工程
- 混淆：破坏代码结构　　增加逆向工程的成本

Obfuscator-LLVM：

- LLVM的开源项目（部分开源）
- 对中间代码进行混淆，与前后端分离
- 主要包括三个PASS

OLLVM中使用的技术

- 指令替换（Instructions Substitution）[-mllvm –sub]
- 伪造控制流（Bogus Control Flow） [-mllvm –bcf]
- 基本块拆分（Basic-Block Splitting）
- 控制流扁平化（Control Flow Flattening）[-mllvm -fla]
- 过程合并（未开源） （Procedures Merging）
- 代码防篡改（未开源） （ Code Tamper-Proofing）

OLLVM中使用的技术——指令替换

- 使用等效但更复杂的表达式进行替换
- 只替换整数运算和布尔运算
- 浮点运算的替换会引入误差
- 很容易被优化
- 加入代码多样化
  - **Bringing Code Diversification**
  - 同一源程序多次编译结果不相同
  - 随机进行指令替换
  - 安全的伪随机数生成器

Operator	Equivalent Instruction Sequence

`a = b + c`	`a = b - (-c)`
	`a = -(-b+ (-c))`
	`a = b + r; a += c; a -= r`
	`a = b - r; a += c; a += r`

`a = b - c`	`a = b + (-c)`
	`a = b + r; a -= c; a -= r`
	`a = b - r; a -= c; a += r`

`a = b & c`	`a = (b^ !c) & b`
`a = b | c`	`a = (b&c) | (b^ c)`
`a = b ^ c`	`a = (!b&c) | (b&!c)`

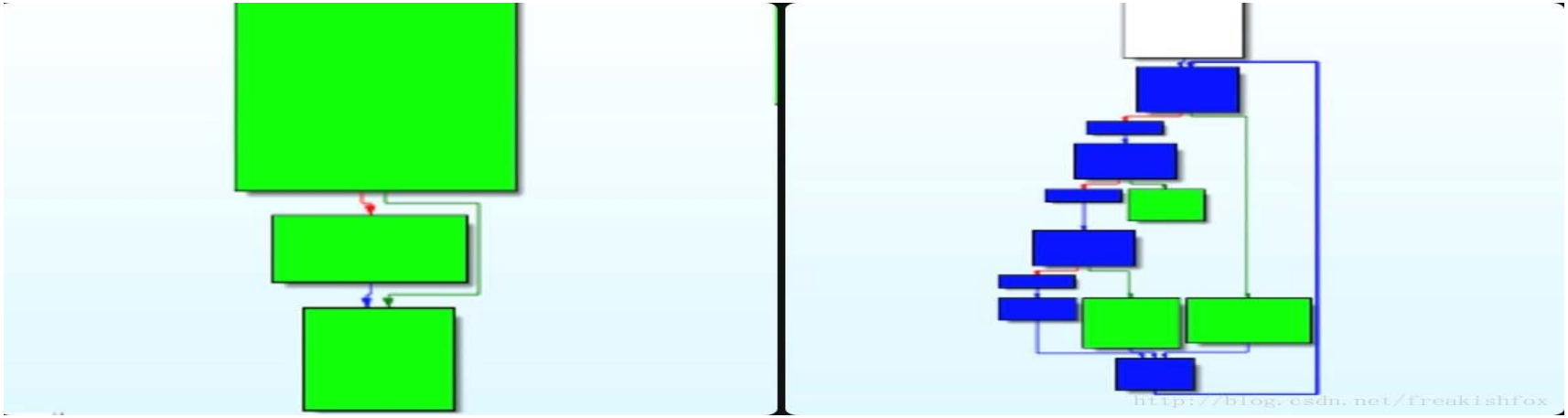
OLLVM中使用的技术——伪造控制流

- 插入多个条件分支跳转
- 可以设置插入的密度（basic block被混淆的几率）
  - boguscf-prob
  - 默认30%
- 可以设置迭代运行的次数
  - boguscf-loop
  - 默认1次
- 不透明谓词
  - 使用值相同但难以逆向（优化器无法识别）的谓词

OLLVM中使用的技术——控制流扁平化

- 思路：破坏原本的控制流——用switch替换if

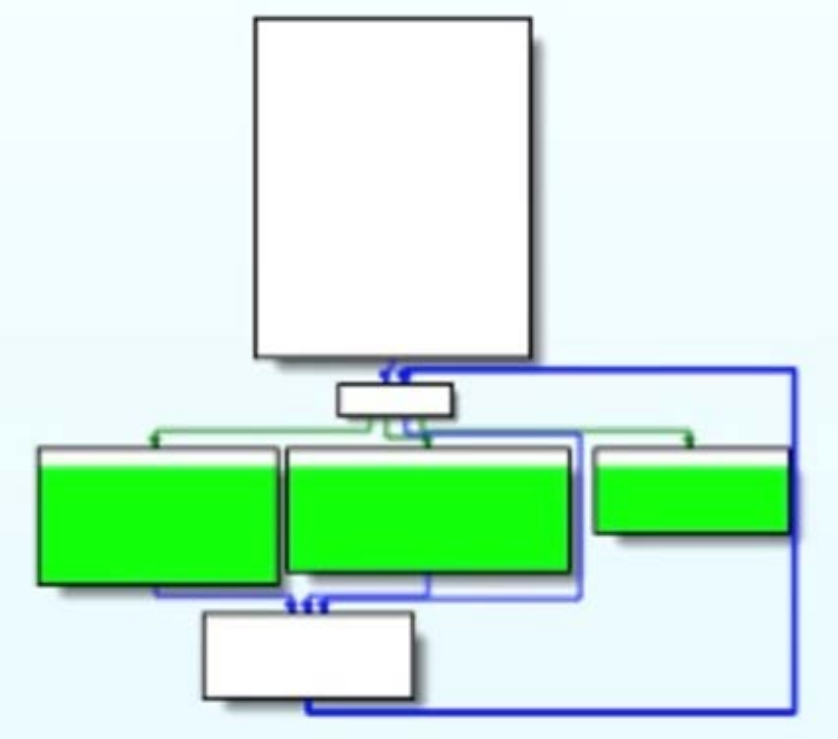
1. 搜集所有基本代码块（Basic Block）
2. 把基本代码块放到控制流图最底部，删除原跳转关系
3. 添加控制流分发逻辑，还原之前的跳转关系



OLLVM中使用的技术——控制流扁平化

4. 在原本的基本块末尾添加“路由变量”，使正确执行

- 通过-mllvm –splitNum开启基本块拆分，进一步破坏代码结构



LLVM测试框架（LLVM Test-suite）

· 测试套件

测试套件包含完整程序，程序代码通常用高级语言如C/C++，可以编译链接进某个可执行程序。这些程序有用户特定编译器编译，然后执行捕获程序输出和时序信息。这些输出和参考输出比较以确保程序编译正确。

除了编译和执行程序，完整程序测试还可作为LLVM性能基准，包括衡量程序生成效率和LLVM编译速度、优化和代码生成。

SPEC 2006

SPEC CPU 2006 benchmark是CPU测试基准套件

包括INT基准和fp基准测试用例

可以指定测试数据集的大小

- -i 参数 ref 最大 Train中等 test最小

使用LLVM测试框架运行SPEC2006对OLLVM进行性能测试

- 获取一个SPEC2006；下载并编译OLLVM和LLVM测试套件
- 设置测试用例（对照组 Baseline）[通过测试套件的CMAKE]
  - 设置编译器路径（ OLLVM /bin/clang）
  - 设置SPEC2006路径
  - 这是构建套件子路径（只选择External）
  - 设置SPEC的运行模式为ref（最大测试数据集）
- 编译测试用例
- 测试并输出结果`$llvm-lit -v -j 1 -o baseline.json .`

使用LLVM测试框架运行SPEC2006对OLLVM进行性能测试

- 设置测试用例（实验组）[通过测试套件的CMAKE]
  - 设置编译器路径（ OLLVM /bin/clang）
  - 设置额外的编译选项`-mllvm -sub -fla -bcf`开启混淆
  - 设置SPEC2006路径
  - 这是构建套件子路径（只选择External）
  - 设置SPEC的运行模式为ref（最大测试数据集）
- 编译测试用例
- 测试并输出结果`$llvm-lit -v -j 1 -o result.json .`

03 OLLVM-性能评估

使用LLVM测试框架运行SPEC2006对OLLVM进行性能测试

- 对比结果`$test-suite/utils/compare.py baseline.json result.json`

Program	未混淆	混淆	差异
400.perlbench.test	492.14	11109.9	2157.50%
401.bzip2.test	590.39	5265.76	791.90%
403.gcc.test	361.88	4655.78	1186.50%
429.mcf.test	466.78	1372	193.90%
433.milc.test	521.03	1276.8	145.10%
445.gobmk.test	573.19	9370.49	1534.80%
450.soplex.test（fp）	293.26	294.08	0.30%
453.povray.test（fp）	212.41	208.46	-1.90%
456.hmmer.test	518.5	6027.5	1062.50%
462.libquantum.test	424.21	6809.5	1505.20%
464.h264ref.test	695.16	4759.63	584.70%
470.lbm.test（fp）	417.02	554.41	32.90%
471.omnetpp.test	423.26	426.01	0.70%
482.sphinx3.test（fp）	584.48	5530.34	846.20%
483.xalancbmk.test	314.32	310.8	-1.10%

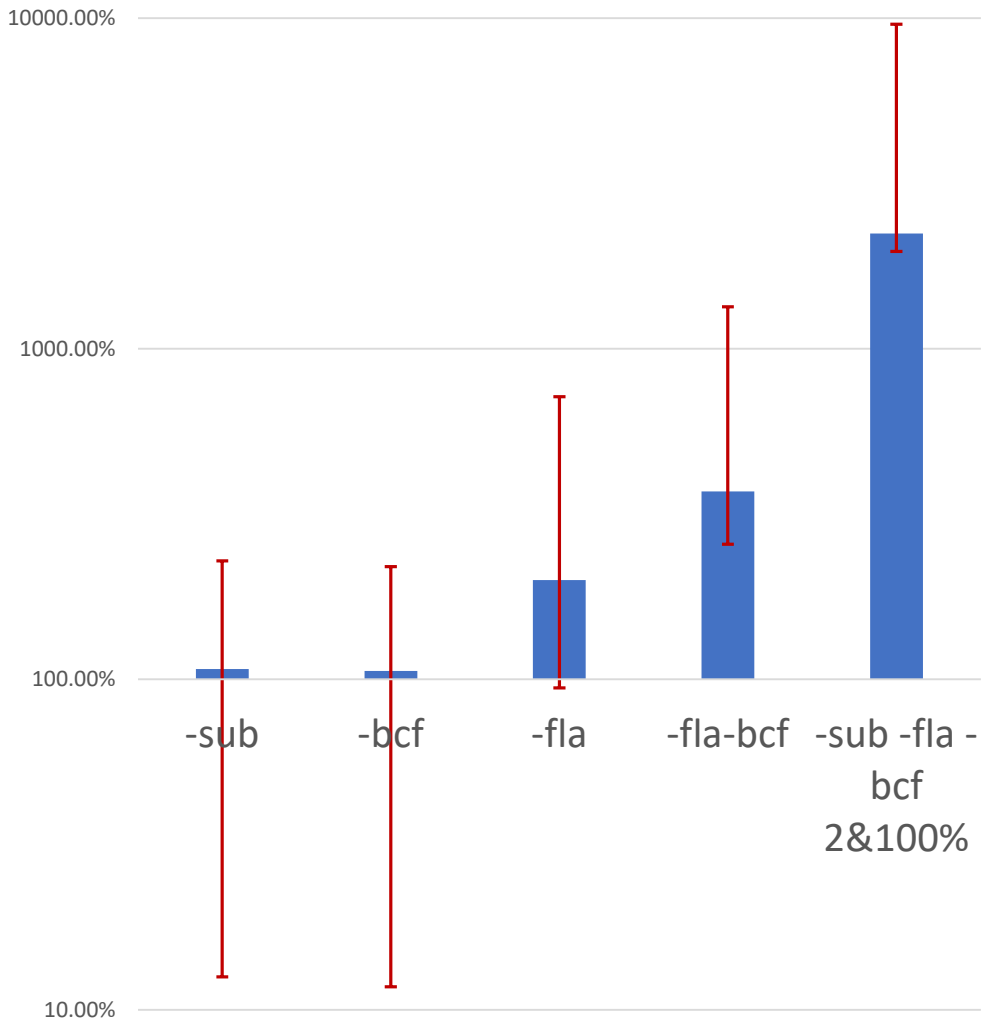
最大	2157.50%
平均	669.28%
最小	-1.90%
方差	45.5898

03 OLLVM-性能评估

性能

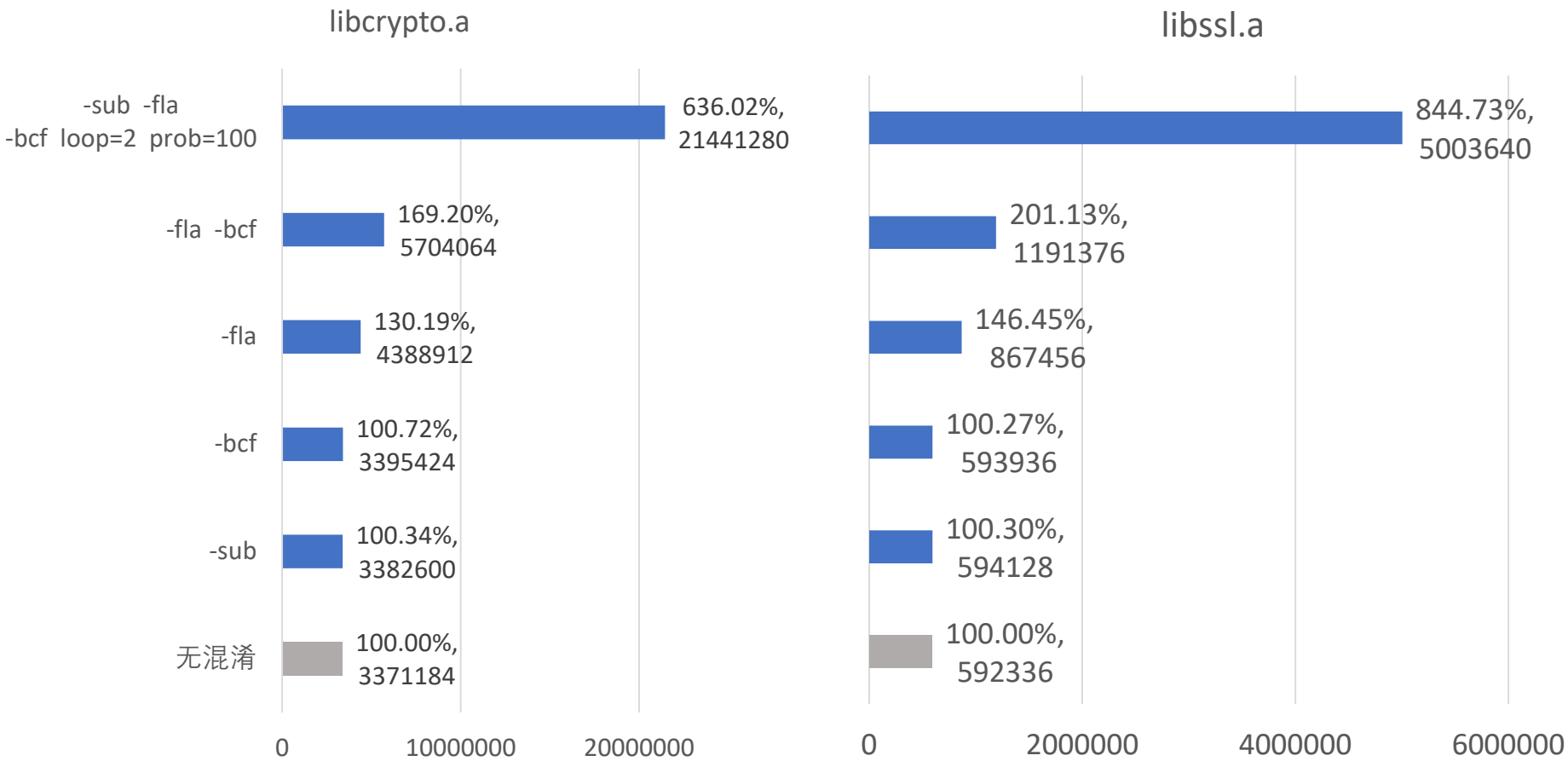
官方给出测试结果：

	均值	方差	最大	最小
-sub	107.35%	0.00239	120.65%	94.77%
-bcf	105.88%	0.00085	113.28%	94.14%
-fla	199.50%	0.85012	516.99%	105.38%
-fla-bcf	370.38%	6.47654	970.78%	114.45%
-sub -fla -bcf 2&100%	2230.17%	390.90	7362.26 %	258.42%



代码体积

官方给出测试结果：



谢 谢

2019/12/18

04 资源

- **为OLLVM添加字符串混淆功能——**上海交大GoSSIP小组
  - https://github.com/GoSSIP-SJTU/Armariris
  - **Apache许可证**
  - Win10 OLLVM添加字符串混淆踩坑篇 https://zhuanlan.zhihu.com/p/39322683
- **Obfuscator-llvm源码分析**
  - https://zhuanlan.zhihu.com/p/39479793