编译大赛
经验谈

2021编译系统设计赛
第一次技术培训

邢其正 @ 北京大学

第一届编译大赛

2020年8月20日

第一届编译大赛

21支队伍进入决赛圈

传统强校仍冲锋在前

中科大上演惊天逆转

我默默仰视各路神仙


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
躺平就好了


[redacted]
科大牛逼


[redacted]
科大牛逼


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了


[redacted]
躺平就好了

编译大赛难吗?

参赛需要具备什么条件

编译大赛难吗?

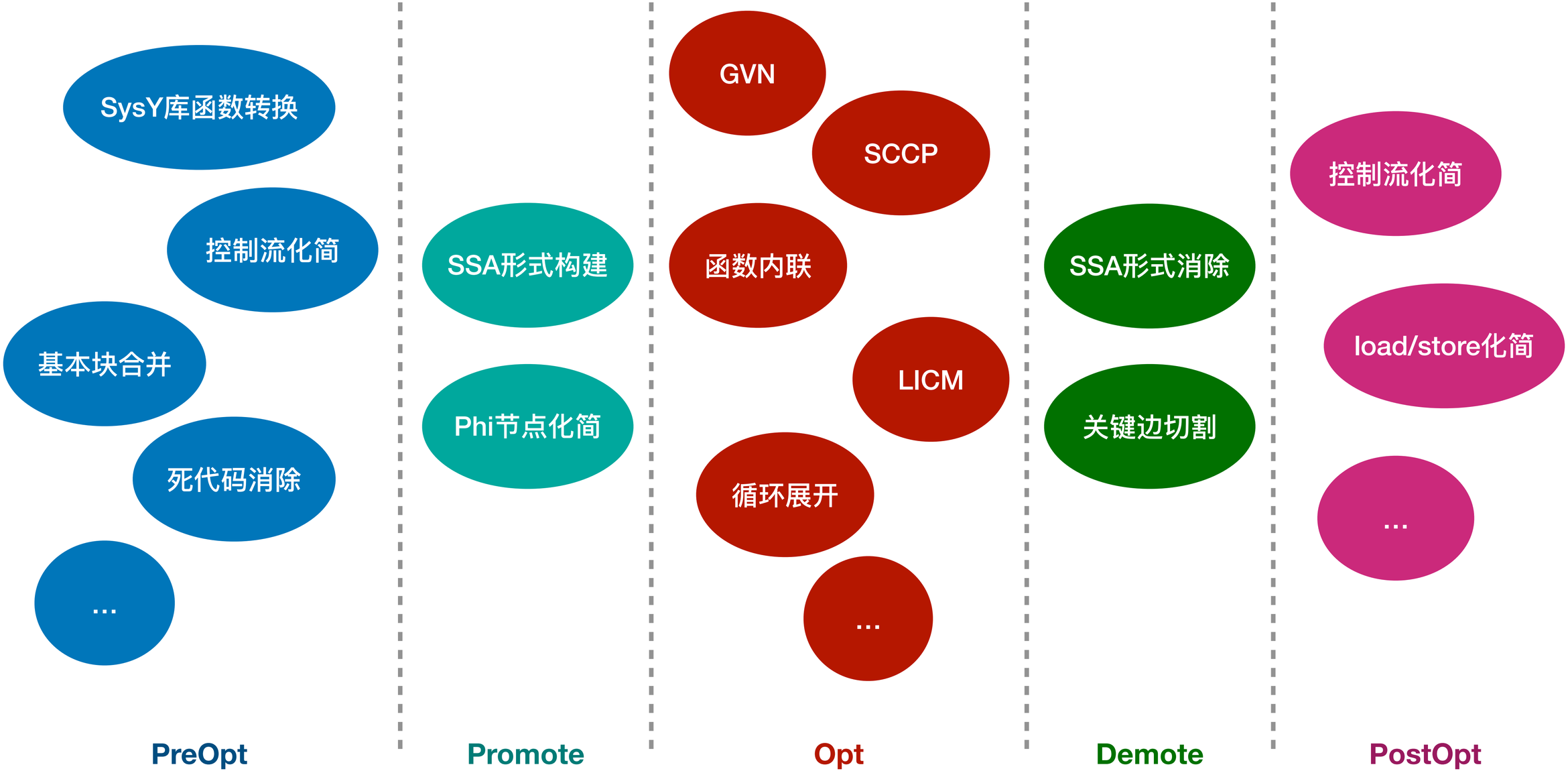
- 实现一个**SysY到汇编**的编译器并不难
- 通过**所有功能测试**即可进入决赛
- 一些简单的优化可**显著提升性能**
- 脚踏实地，**逐步求精**

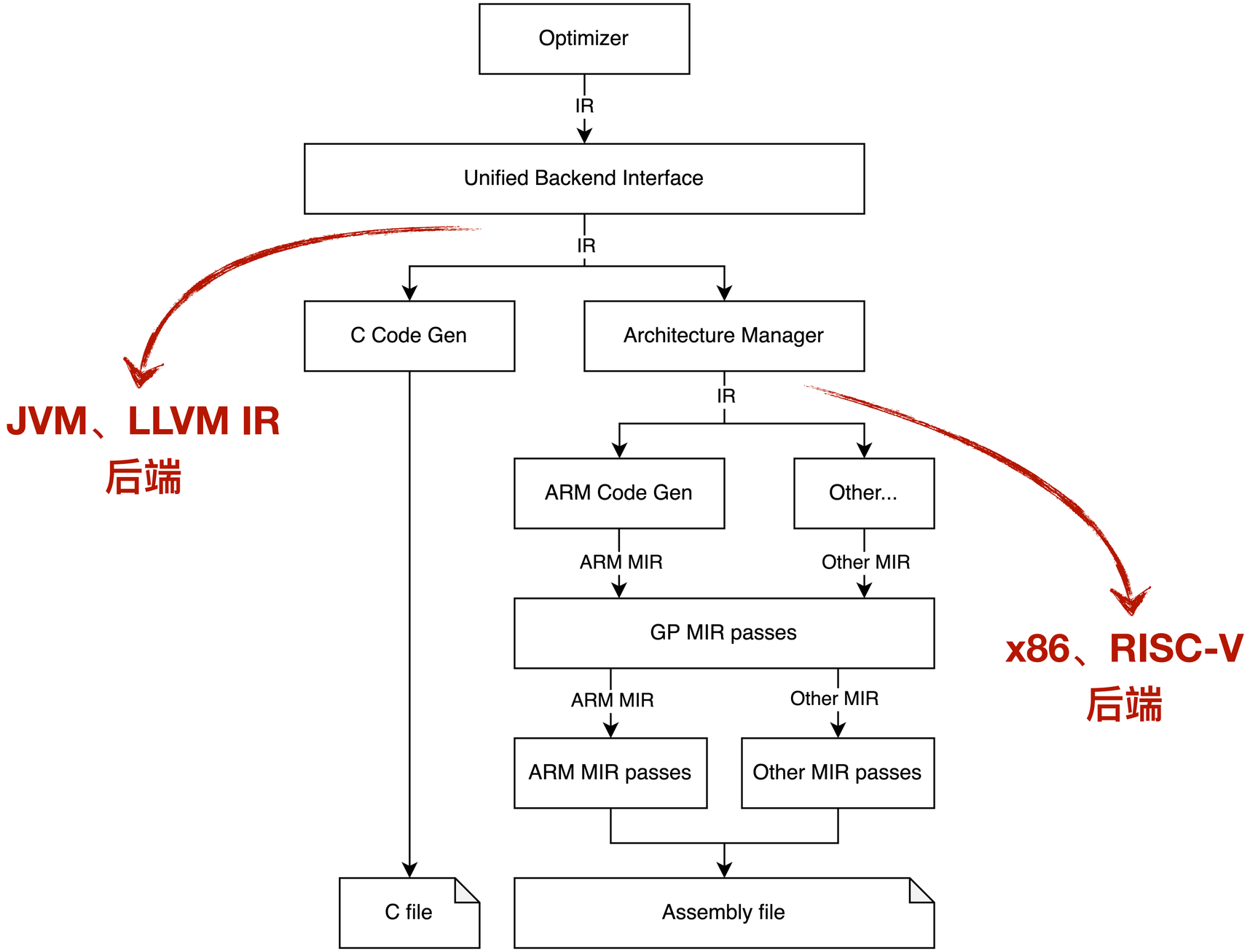
我们做了什么？

北京科技大学参赛队的作品：MimiC

MimiC

- **手写**的编译器前端
- 贯穿前后端的**静态类型检查**系统
- **模块化**的Pass和**Pass管理器**
- 基于**SSA形式**的**强类型**IR
- 支持**多个目标平台**的代码生成器


SysY库函数转换
控制流化简
基本块合并
死代码消除
...
PreOpt
SSA形式构建
Phi节点化简
Promote
GVN
SCCP
函数内联
LICM
循环展开
...
Opt
SSA形式消除
关键边切割
Demote
控制流化简
load/store化简
...
PostOpt


Optimizer
IR
Unified Backend Interface
IR
C Code Gen
Architecture Manager
IR
ARM Code Gen
Other...
ARM MIR
Other MIR
GP MIR passes
ARM MIR
Other MIR
ARM MIR passes
Other MIR passes
C file
Assembly file
JVM、LLVM IR
后端
x86、RISC-V
后端

MimiC

已于GitHub开源

完整的**带优化的编译器**实现

不使用除C++标准库外的**任何依赖**

第一届编译大赛**参赛作品**，排行第七

将SysY语言**编译到ARM或RISC-V**

参赛经验分享

抛开项目和源码，能不能讲点干货？

参赛经验分享

方法篇　技术篇

参赛经验分享

方法篇

如何用好C++?

- 良好的**代码风格和注释**：Google Style
- 使用**类和继承**来处理AST、IR等结构
- 智能指针：**shared_ptr和unique_ptr**
- **C++17的新特性**：optional、string_view
- 使用**Sanitizer**检查程序的行为：ASan

如何用好C++?

- 良好的**代码风格和注释**：Google Style
- 使用**类和继承**来处理AST、IR等结构
- 智能指针：**shared_ptr和unique_ptr**
- **C++17的新特性**：optional、string_view
- 使用**Sanitizer**检查程序的行为：ASan

如何用好Java?

用就完事了

使用Git协作

- 入门：learngitbranching.js.org
- 划分公共的分支和私有的分支
- 禁止使用任何强制操作更新公共分支
- 合理使用merge和rebase
- 定义提交、推送和合并的相关规范

重视测试和记录

- 持续使用**各类测例**测试编译器
- 让你的编译器**输出C或LLVM IR**
- 使用**QEMU (Static)**完成跨架构测试
- 编写**自动化测试脚本**自动完成测试
- **使用CI**将测试融入版本控制

参赛经验分享

技术篇

如何设计中间表示(IR)?

- 采用**何种形式**：寄存器式实现简单，**但SSA形式容易优化**
- 是否**分层**：分层利于**保留不同级别的信息**，但复杂
- 使用**何种结构**：树形？线性？图状？(PDG/Sea-of-Nodes)
- 是否**具备类型**：强类型IR保留更多信息，便于优化

SSA形式

Static Single Assignment Form

关于 SSA形式

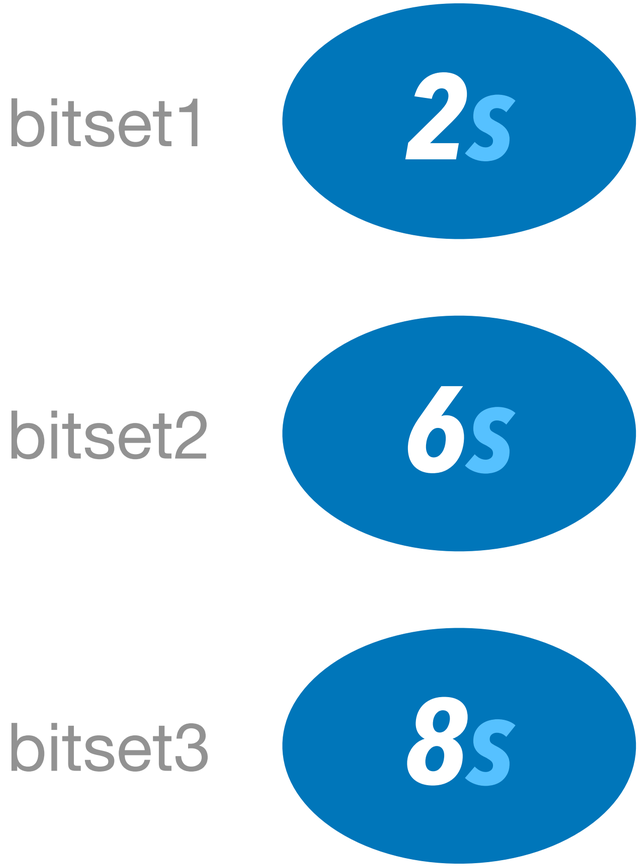
- 如何构造：**支配边界(eager)，值标号(lazy)**
- 如何退出：**切割关键边**，插入load/store
- Phi节点：在执行优化时**不破坏Phi的性质**
- UD/DU链：使用**双向引用结构**存储

怎么做优化?

一个例子：MimiC对bitset测例的优化过程

基线性能

bitset1　　bitset2　　bitset3

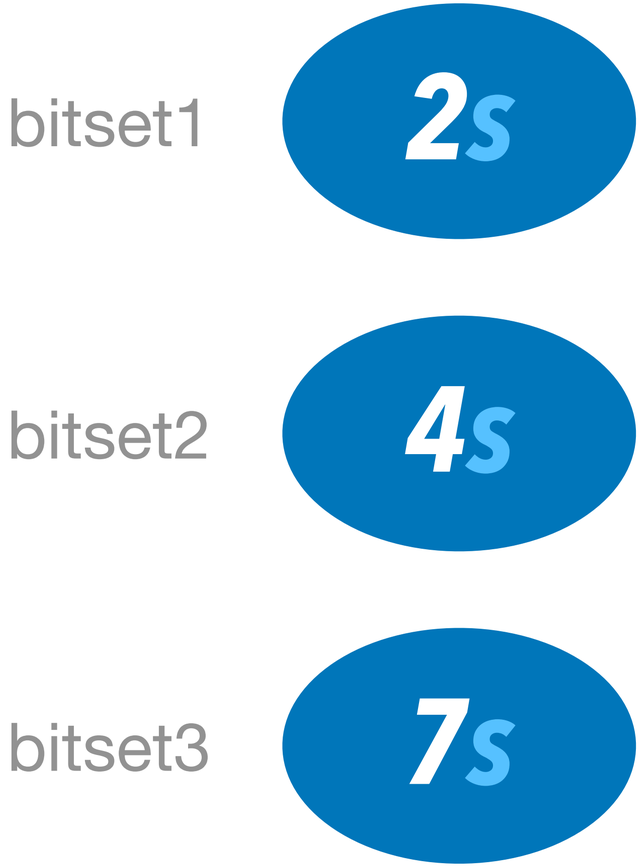


使用硬件除法指令

比赛所使用的硬件平台是**Raspberry Pi 4B**

CPU为一颗**四核心的Cortex-A72**

支持**硬件除法**，所以无需调用库函数



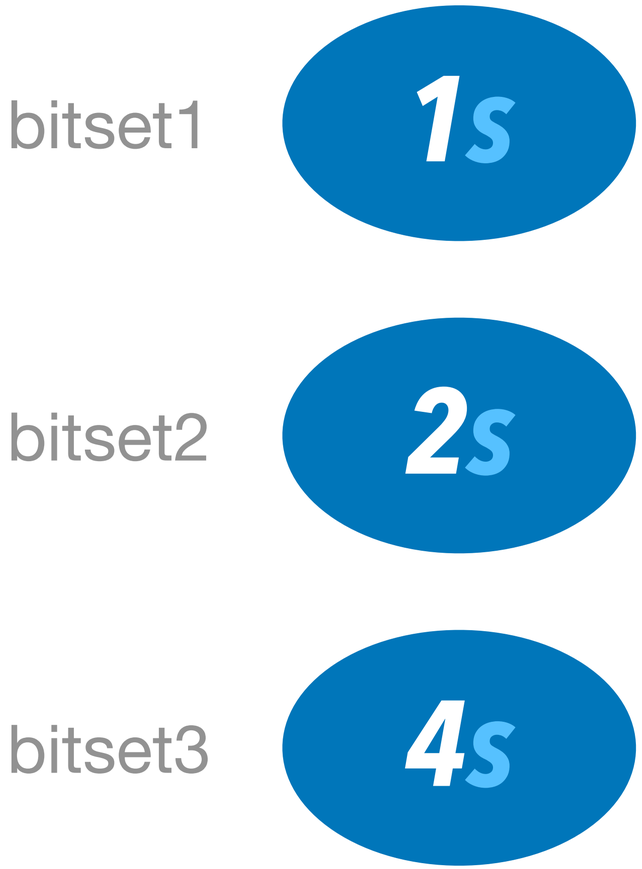
2.7x

使用全局值标号

消除了一部分**冗余的计算**

但程序**只快了一点点**，性能提升不明显

似乎对mm测试用例提升幅度较大



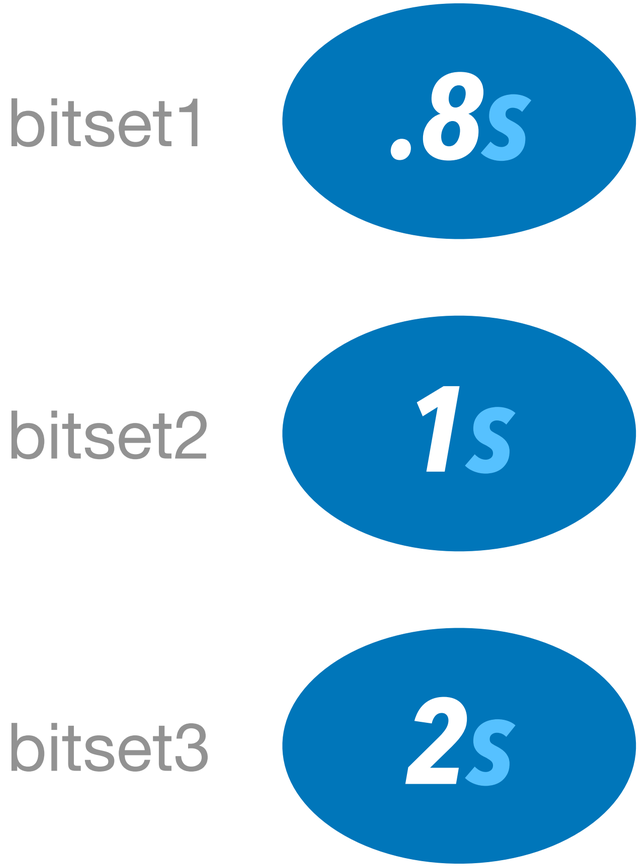
5x

展开次数固定的循环

bitset中使用了一个**循环次数固定的while**

实现一个**简单的循环展开算法**进行模式匹配

性能提升幅度**十分显著**



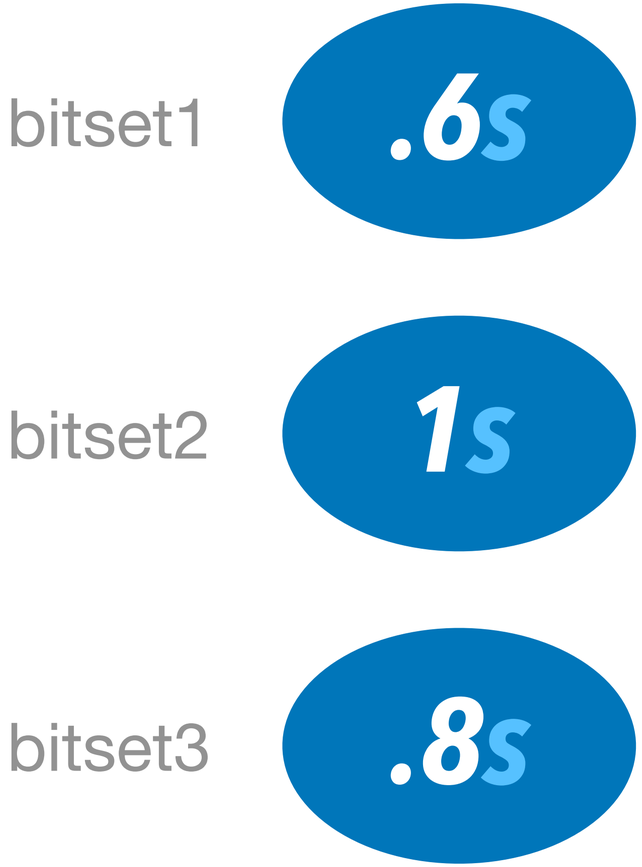
9.2x

优化数组写入操作

bitset中会初始化一个数组的所有元素

之前的SCCP和GVN已经求出了大部分常量

直接将这一系列赋值优化为单个store



14x

将局部数组移到全局

bitset中**每次对set的调用**都会初始化数组

识别所有**只被初始化一次**的局部数组

将其提升到全局，**省掉初始化操作的开销**

关于优化的参考资料

- 书籍：**EAC (英文版)，SSA Book，鲸书，现代体系结构的优化编译器**
- 论文：SSA构造，GVN/GCM，寄存器分配，c9x.me/compile/bib
- 网络文章：知乎上**@RednaxelaFX**的大部分回答和专栏文章
- 开源项目：**LLVM早期版本(2.x)**的源代码，上届大赛的开源代码
- 呼吁参赛者**开放自己的答辩PPT**，赛方可以组织**参赛者进行技术分享**

预祝大家

参赛顺利

謝謝