

Анализ и улучшение алгоритма расположения  
инвариантов цикла в компиляторной  
инфраструктуре LLVM с использованием  
статического и динамического профилирования  
бакалаврская работа

Найданов Евгений Максимович

2023

# Варианты расположения инварианта цикла

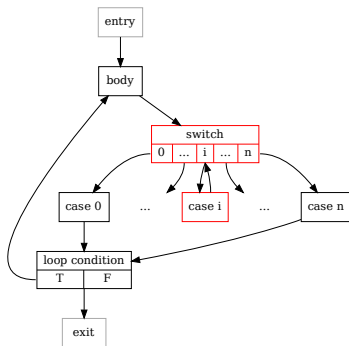
- ▶ **В теле цикла**  
Исходное расположение.
- ▶ **В предзаголовке цикла**  
Оптимально, если число итераций больше единицы.
- ▶ **В некоторых блоках выхода цикла**  
Доступно только если нет использований в цикле.

# Пример: SPEC CPU® 2017. perlbench\_r. S\_regmatch

Структура функции:

- ▶ Основной цикл является горячим.
- ▶ Большое число случаев внутри конструкции switch.
- ▶ Некоторые из них осуществляют безусловный переход в начало тела цикла.

На данном примере при компиляции с использованием Clang, среднее число инструкций на итерацию цикла больше, чем у компилятора GCC.



# Формальная постановка задачи

- ▶  $M \subset \{ d : p \text{ dom } d \}$
- ▶  $\forall u \in U(i) \exists m \in M : m \text{ dom } u$ , где  $U(i)$  - множество блоков, содержащих использование инварианта  $i$ .
- ▶  $\sum_{m \in M} f(m) \rightarrow \min$ , где  $f(m)$  - оценка частоты вероятности исполнения блока  $m$ .

# Подход к размещению инвариантов цикла в компиляторной инфраструктуре LLVM

- ▶ Вынос инвариантов из цикла
- ▶ Пропагация инвариантов в цикл
  - ▶ Анализ частот исполнения блоков
  - ▶ Анализ вероятности переходов

# Вынос инвариантов из цикла

- ▶ Удаляется, если нет использований.
- ▶ Выносится в блоки выхода, если все использования вне цикла.
- ▶ Иначе выносится в предзаголовок.

# Построение анализа частот исполнения блоков

- ▶ Обход циклов – сначала все вложенные циклы, затем внешний.
- ▶ Распределение веса в циклах

$$w_j = \sum_{E(V_i, V_j)} w_i p_{ij}$$

- ▶ Расчет числа итераций циклов

$$s = \frac{1}{w_0 - w_b}$$

- ▶ Распределение веса в функции
- ▶ Расчет частот исполнения блоков

# Построение анализа вероятности переходов

- ▶ Метаданные
- ▶ Эвристика основанная на весах блоков
- ▶ Эвристики основанные на свойствах сравнений



# Эвристика основанная на весах блоков

- ▶ Присвоение малых весов ребрам, входящим в редко исполняемые блоки.
- ▶ Для ребер, с условием, которое перестает выполняться после некоторой итерации – вдвое меньший вес.
- ▶ Веса дуг, выходящих из цикла, делятся на оценку числа итераций цикла.

# Эвристики основанные на свойствах сравнений

| Эвристика                          | Условие                | Вероятность истинности условия |
|------------------------------------|------------------------|--------------------------------|
| Сравнение указателей               | $p1 == p2$             | 0.375                          |
|                                    | $p1 != p2$             | 0.625                          |
| Сравнение с константой             | $i == 0$               | 0.375                          |
|                                    | $i < 0$                |                                |
|                                    | $i == -1$              |                                |
|                                    | $i <= 0$               |                                |
|                                    | $i != 0$               | 0.625                          |
|                                    | $i > 0$                |                                |
|                                    | $i != -1$              |                                |
|                                    | $i >= 0$               |                                |
| Сравнение чисел с плавающей точкой | $f1 == f2$             | 0.375                          |
|                                    | $f1 != f2$             | 0.625                          |
|                                    | <code>!isnan(f)</code> | $1 - 2^{-20}$                  |
|                                    | <code>isnan(f)</code>  | $2^{-20}$                      |

# Получение динамического профиля исполнения программы

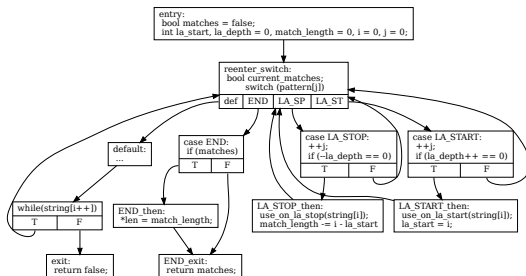
- ▶ Инструментация кода.
- ▶ Запуск программы для сбора профиля.
- ▶ Использование профиля при компиляции программы.

# Сравнение с подходом в компиляторной инфраструктуре GCC

- ▶ Вынос инвариантов цикла не является приведением к канонической форме
  - ▶ Каждый инвариант обрабатывается строго один раз.
  - ▶ Больше инструкций в теле цикла.
- ▶ Статический анализ частот исполнения блоков точнее:
  - ▶ Использование большего количества эвристик.
  - ▶ Использование алгоритма объединения эвристик.

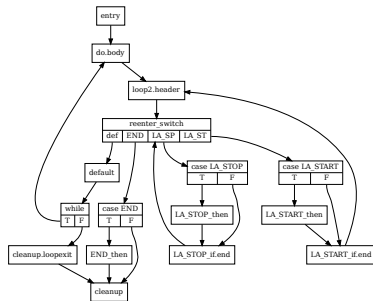
# Упрощенный пример S\_regmatch\_draft

- ▶ Схожая структура
- ▶ `string[i]` – инвариант для внутренних циклов, но не для внешнего.



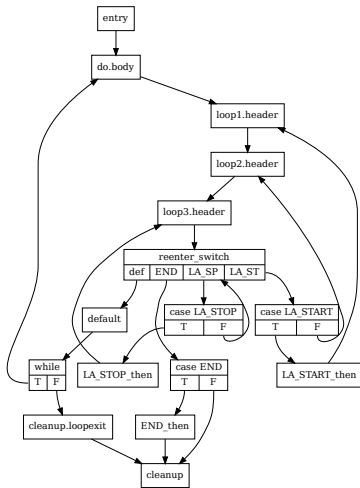
# S\_regmatch\_draft. Построение упрощенной формы циклов

- ▶ Предзаголовок цикла `do.body`.
- ▶ Заголовок `loop2.header`.



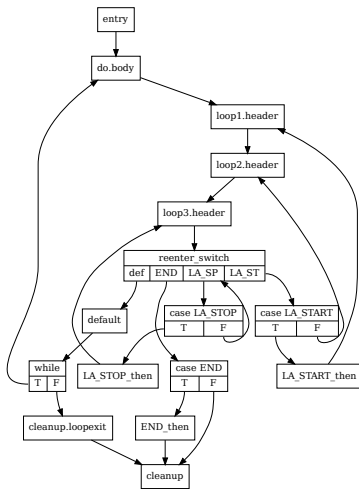
## S\_regmatch\_draft. Упрощенная форма циклов

- ▶ Заголовки loop1-3.header и reenter\_switch.
- ▶ Предзаголовки do.body и loop1-3.header.



# Расположение инварианта `string[i]`

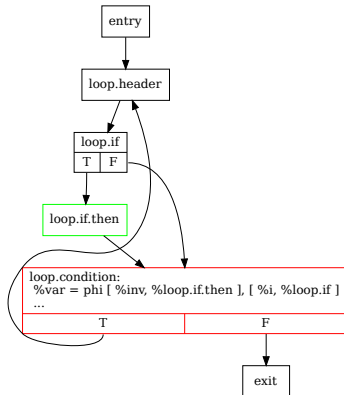
- ▶ Вынос инвариантов из цикла – расположен в `do.body`.
- ▶ Если не пропагирован, число исполнений – длина строки `string`.
- ▶ Если спropaгирован в `LA_START_then` и `LA_STOP_then` – сумма числа токенов `LA_START` и `LA_STOP` в шаблоне `pattern`.





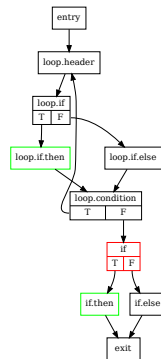
# Предлагаемые улучшения. Пропагация инвариантов с использованием в $\varphi$ узлах

- ▶ Многие инварианты цикла могут быть использован в некотором  $\varphi$  узле.
- ▶ Невозможно пропегировать в базовый блок, в котором находится этот  $\varphi$  узел.
- ▶ Решение: рассматривать базовый блок из которого поток управления входит в узел с данным значением.



# Предлагаемые улучшения. Пропагация инвариантов во все доминируемые базовые блоки

- ▶ Использование инварианта как внутри, так и вне цикла.
- ▶ Рассматривать блоки цикла и блоками выхода недостаточно.
- ▶ Следует рассматривать все базовые блоки, доминируемые предзаголовком.



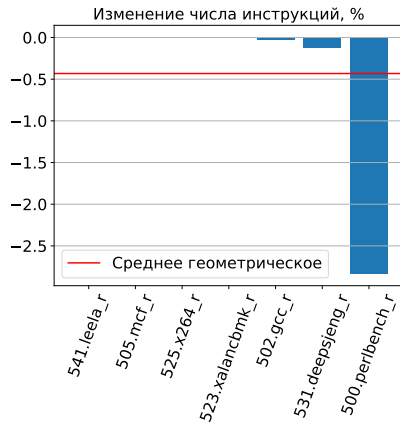
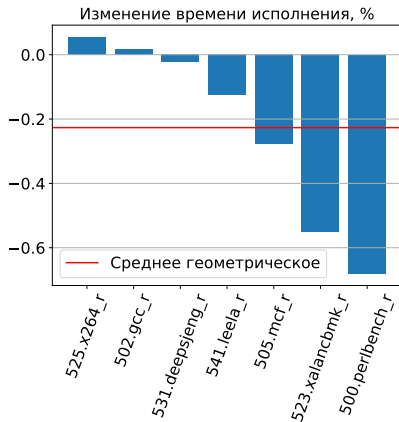
# Анализ полученного алгоритма

- ▶ В работе доказана оптимальность получаемых расположений с точки зрения минимизации суммарной частоты исполнения.
- ▶ Асимптотика алгоритма:
  - ▶ В среднем:  $O(N)$
  - ▶ В худшем случае:  $O(N^2)$

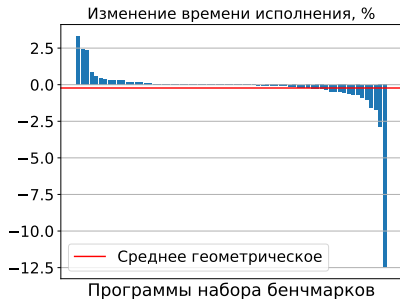
# Производительность алгоритма. Методика измерений

- ▶ СнК Alibaba T-Head XuanTie C910 ICE, Линукс. 2 ядра. 1.2 ГГц. RISC-V 64 GC.
- ▶ Время исполнения программы в тактах и число исполненных инструкций
- ▶ Сбор значений соответствующих счетчиков процессора, при помощи программы perf.
- ▶ Использовалось динамического профилирование.

# Производительность алгоритма. SPEC CPU® 2017



# Производительность алгоритма. Коллекция тестов LLVM



# Заключение

- ▶ Показана необходимость улучшения алгоритма расположения инвариантов цикла в компиляторной инфраструктуре LLVM.
- ▶ Проведен общий анализ алгоритма, и анализ его применения к функции `S_regmatch`.
- ▶ Разработаны улучшения алгоритма пропагации инвариантов в тело цикла.
- ▶ Для улучшенного алгоритма, была доказана оптимальность получаемого расположения инвариантов.
- ▶ Алгоритм был реализован в компиляторной инфраструктуре LLVM.
- ▶ Анализ производительности:
  - ▶ Значительное увеличение для некоторых приложений
  - ▶ Прирост производительности в среднем.
- ▶ Изменения включены во внутреннюю поставку компилятора компании Синтакор.

Спасибо за внимание!