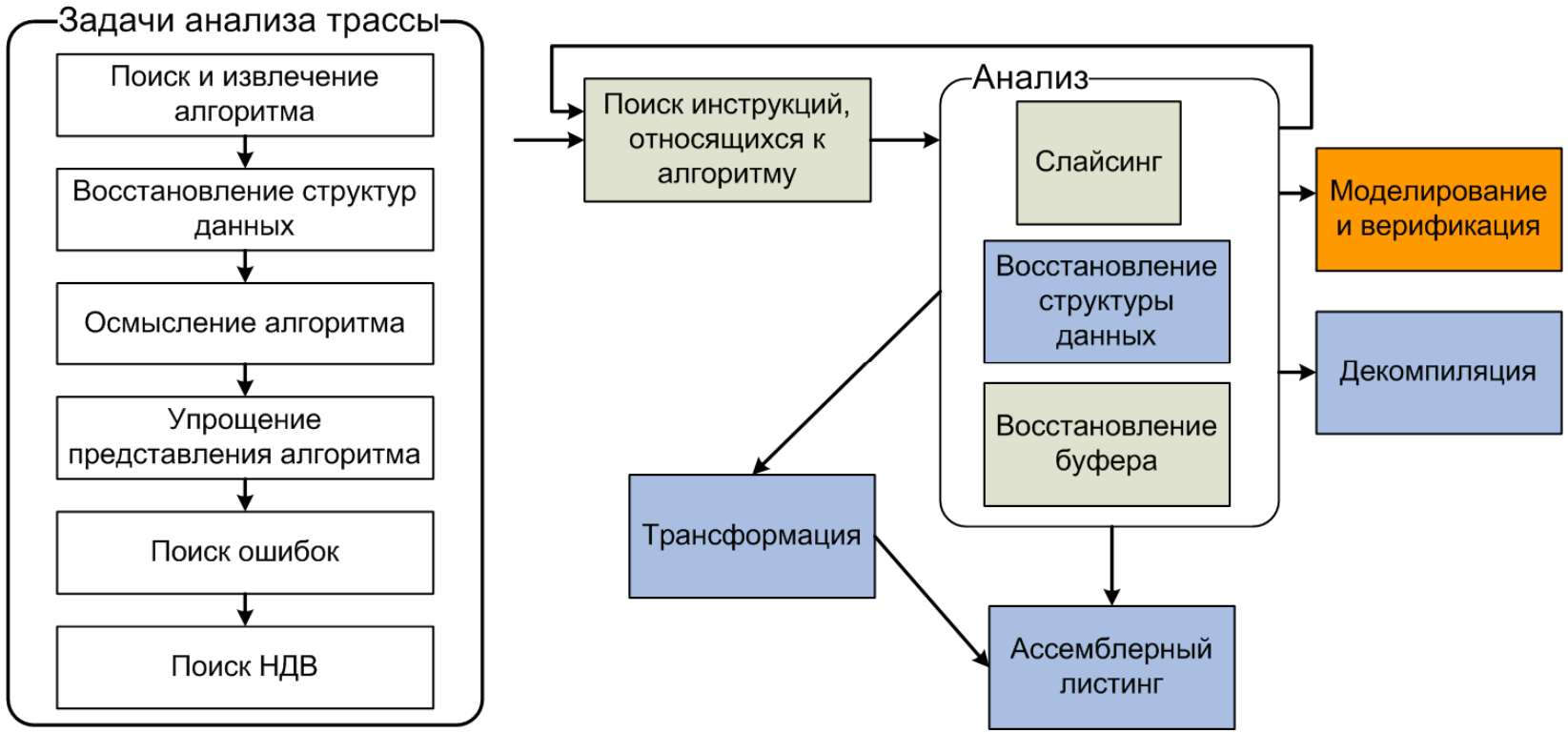
Моделирование семантики машинных инструкций

Падарян В.А., Соловьев М.А.
{vartan, eyescream}@ispras.ru
Институт системного программирования РАН
http://www.ispras.ru

РусКрипто'2010, 3 апреля 2010 г.

Методика анализа



Ключевые особенности среды анализа

- Развитая инфраструктура. Различные целевые платформы. Единая БД для хранения результатов анализа.
- Полноценное выявление алгоритмов на уровне машинных инструкций. Минимальные задержки отклика системы.
- Автоматизированное восстановление структур данных.
- Интеграция со средствами статического анализа.
- Декомпилятор

Актуальные направления исследований

- Поддержка многоядерных систем. Отслеживание взаимодействия процессов.
- Распараллеливание алгоритмов анализа.
- Привлечение методов эмуляции и статического анализа для анализа незадействованных при снятии трассы ветвей алгоритма.
- Разработка внутреннего представления для моделирования семантики машинных инструкций.
- Модель программы с возможностью верификации.
  - Восстановление автомата состояний при анализе протокола.

Требования

- Моделирование семантики пользовательских и системных инструкций широкого класса архитектур
- Возможность применения к моделям компиляторных оптимизаций
- Интерпретация моделей

- Использование внешних спецификаций

Обзор работ

- Valgrind
  - Нет поддержки внешних спецификаций.
  - Нет поддержки системных инструкций.
- UQBT, UQDBT
  - Нет поддержки системных инструкций.
- SimpleScalar
  - Нет поддержки CISC-архитектур.
- iDNA
  - Нет поддержки внешних спецификаций.
  - Нет поддержки системных инструкций.

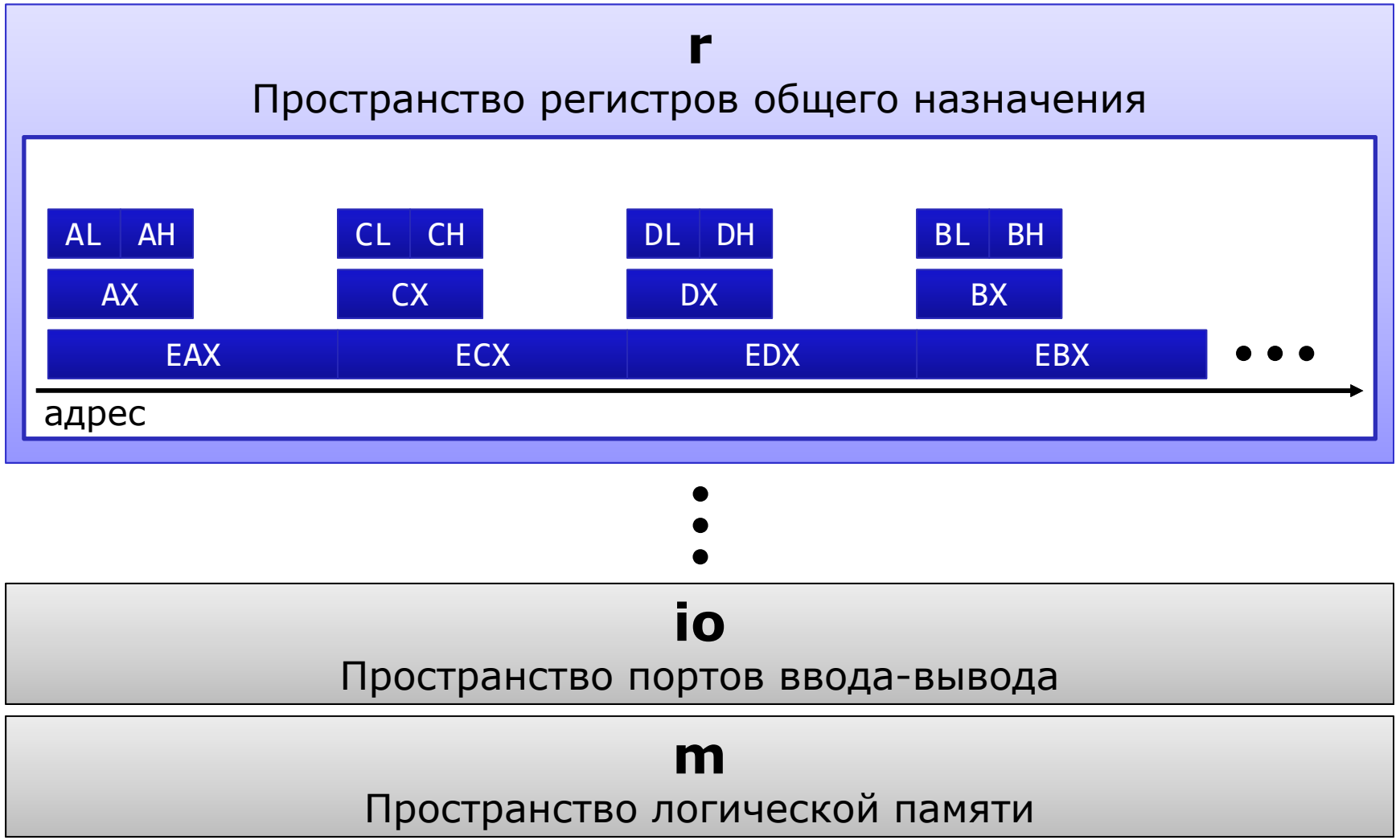
Сложности

- Побочные эффекты в CISC-архитектурах; влияние на флаговое слово.
- Нетривиально организованные регистровые файлы: регистровые окна, теневые наборы.
- Системный код: переключение контекстов, работа с периферией.

Использование модели

Модель машины

Модель адресных пространств



Подстановки

Подстановка

Заголовок подстановки

Тело подстановки

Последовательность операторов

Локальные переменные: t.1, t.2…

Операторы модели

- **NOP**
- **INIT** `t.n : atom = constant`
- **APPLY** `t.n = operation(t.x, t.y, …)`
- **BRANCH** `condition => target`
- **LOAD** `t.n : atom = space:t.a`
- **STORE** `space:t.a = t.n`
- **SPECIAL** `HALT, TRAP #n…`
- **ANNOTATION**

Пример спецификации

```
match "CMP" ref #, val i8 with i16, i32, i64
begin
    /* Compare with extension.  */
    discard sub.#($1, sx.i8.#($2))

    /* Update flags.  */
    r:flags = uf(r:flags, (i16) 0x08D5)
end
```

Пример модели инструкции

CMP EAX, 100h

```
; Подстановка операндов: t.-1 = address-of(EAX), t.-2 = 100h.
INIT t.-1 :i16 = 0x0000
INIT t.-2 :i32 = 0x00000100

; Применение операции вычитания к операндам.
LOAD t.0  :i32 = r:t.-1
APPL t.1  :i32 = sub.i32(t.0, t.-2)

; Обновление флагового слова x86.
INIT t.2  :i16 = 0x40
LOAD t.3  :i16 = r:t.2
INIT t.4  :i16 = 0x08D5
APPL t.5  :i16 = uf(t.3, t.4)
STOR r:t.2     = t.5
```

Перспективы

- Повышение уровня представления семантики программы
- Архитектурно-независимые алгоритмы:
  - декомпиляция;
  - оптимизация.
- Анализ нереализованных путей в семействе связанных трасс
- Повторная интерпретация машинного кода