План

- Методы защиты от анализа
- Декодирование и дизассемблирование
  - Вспомогательные данные, используемые при анализе
- Комбинированный анализ

Методы защиты от анализа

- Навесная
  - Упаковка/шифрование исполняемого файла
    - Файлов полученных в бинарном виде
    - Как последний этап защиты после компиляции из исходных кодов, с учетом дополнительной информации полученной при компиляции
- Встраиваемая
  - Запутывание программного кода во время компиляции
  - Встраивание различных приемов, защиты от тестирования отладчиками
- Виртуальная машина
  - Встраивание интерпретатора инструкций сторонней архитектуры и исполнение на нем частей программы

Методы защиты от анализа

- ”Песочница”
  - Помещение программы в изолированное окружение
- Водяные знаки
  - По бинарной программе, можно однозначно определить ключ пользователя
  - Генерация различного кода программы в зависимости от ключа при одинаковой функциональности
- Привязка программы к ключевому носителю (диск, смарт карты,usb dongle и т.д.)
- Запуск зашифрованной программы только на разрешенном оборудовании (производитель, серийные номера комплектующих)

Антиотладка

- Антиотладка - приемы против отладчиков эксплуатирующая особенности:
  - отладчика
  - операционной системы
  - реализации механизма отладки ОС
  - общие антиотладочные приемы
  - особенности аппаратуры
- Затруднение записи на диск областей памяти процесса

Борьба с программным отладчиком

- Поиск по заголовкам открытых окон
- Поиск по сигнатуре процесса
- Анализ реестра на наличие характерных ключей
- Открытие характерных объектов ядра (например \\.\SICE, \\.\SYSER)
- Эксплуатация уязвимостей отладчика (OllyDbg и OutputDebugString)

Особенности ОС

- Использование специальных API
  - ***isDebuggerPresent***
  - ***NtQueryInformationProcess***
- Проверка полей структур описывающих процесс
  - ***PEB.BeingDebugged***
  - ***EPROCESS->DebugPort***
- Отладчик должен иметь SeDebugPrivilege для отладки системных процессов, но во многих отладчиках эту привилегию получают всегда
- Обработка исключений в отлаживаемой программе
  - В большинстве случаев, при наличии отладчика исключения обрабатываются отладчиком, а не передаются программе

Особенности архитектуры x86

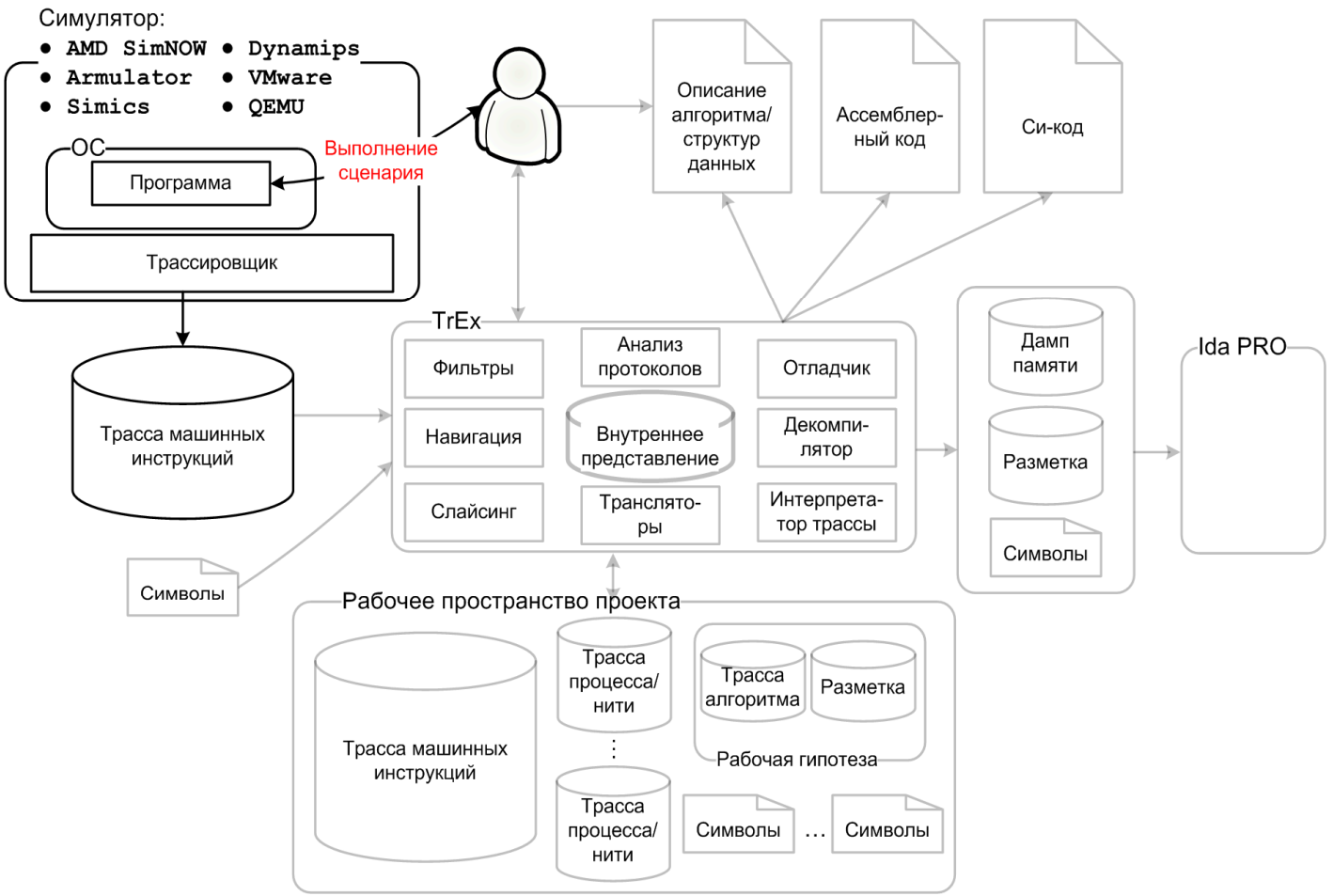
- Перехват прерываний int 1, int 3
- Замер времени выполнения критичного участка кода (RDTSC)
- Проверка контрольных сумм критичных участков кода и атрибутов доступа к страницам
- Очередь предварительной выборки (Prefetch queue)
  - На x86 процессорах до Pentium очередь предварительной выборки не очищалась автоматически, когда происходила запись в адрес памяти, который соответствовал адресу байтов в очереди предварительной выборки. Очередь очищалась всякий раз, когда происходило исключение.
- Селекторы сегментных регистров
  - Значения селекторов выглядят постоянными, но это не так, их значение может быть изменено, но вскоре оно будет установлено в значение по умолчанию. Некоторые события вызывают сброс его значения в значение по умолчанию, одно из таких событий - исключение. В контексте отладки исключение пошаговой трассировки может иметь дополнительный побочный эффект.

Преодоление защиты

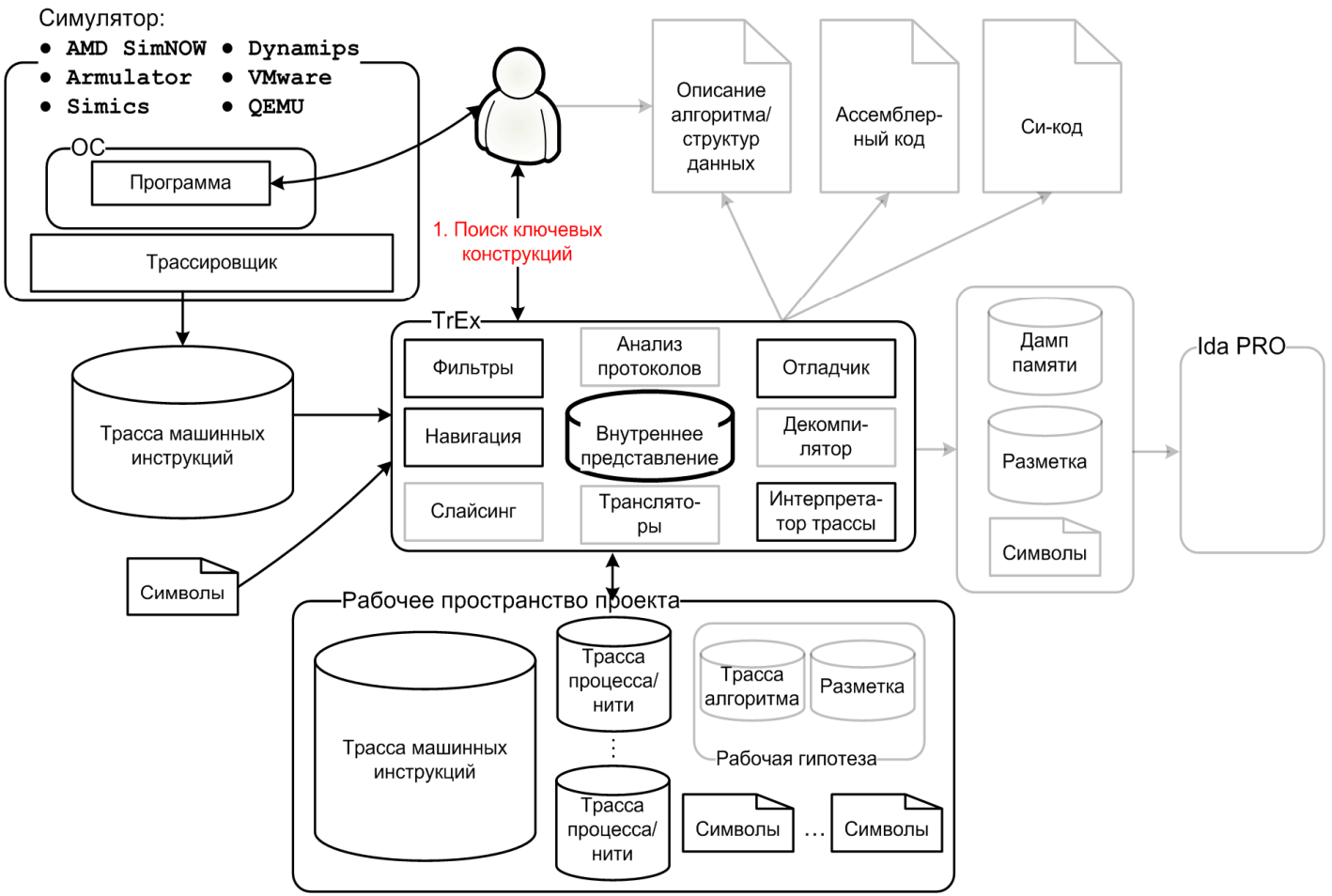
- Плагины к отладчикам, для обхода известных антиотладочных приемов
- Плагины или скрипты для дизассемблеров, для удаления запутывающих преобразований
- Специализированные автоматические или полуавтоматические инструменты направленные на преодоление защиты установленной конкретным инструментом
- Автоматические или полуавтоматические инструменты направленные на преодоление широкого класса защит, например автоматические распаковщики поддерживающие широкий спектр защит.

Для защиты от тестирования отладчиками, требуется как поиск новых уязвимостей в отладчиках, так и поиск новых способов затруднения отладки основанных на использовании особенностей ОС и оборудования

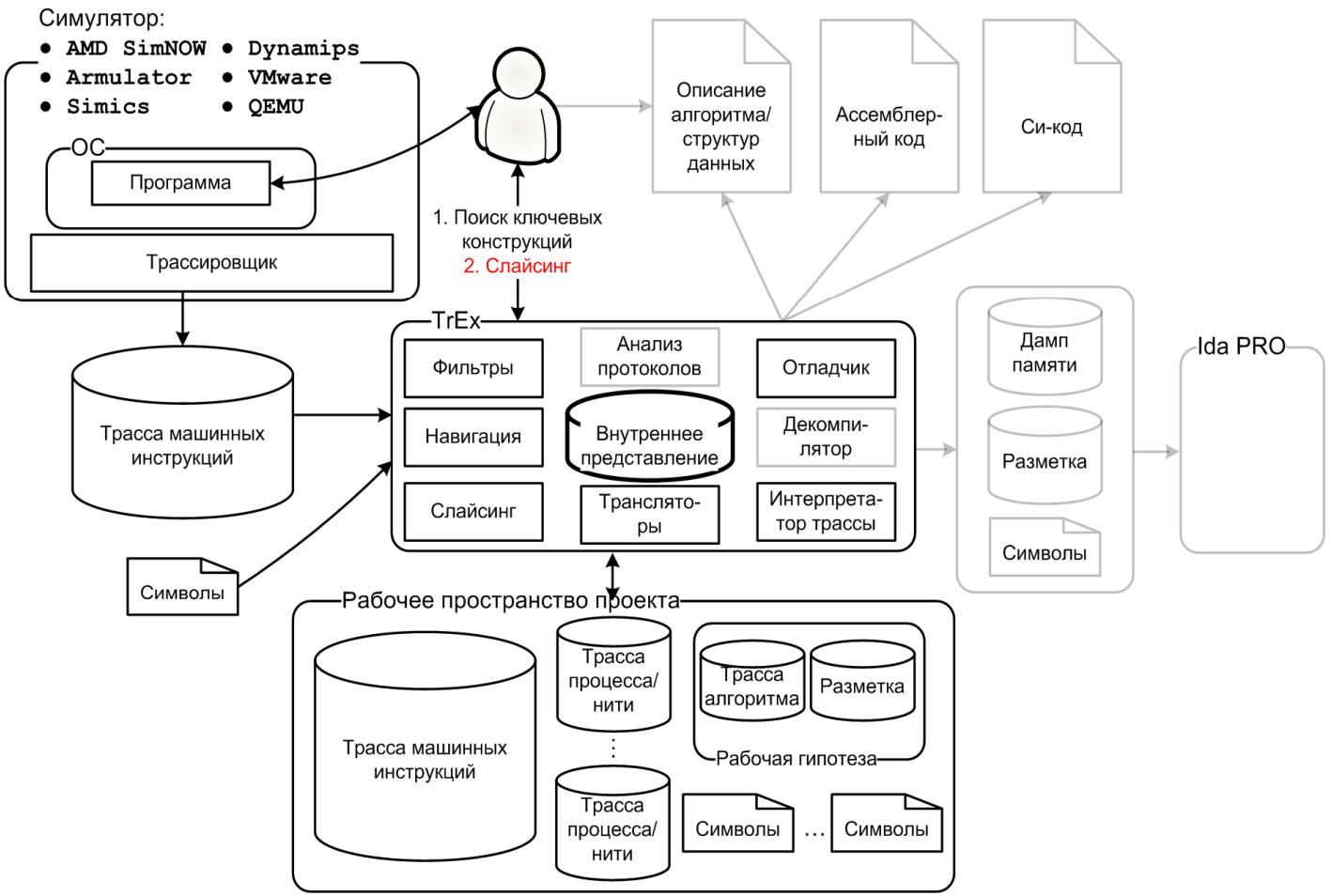
Комбинированный анализ



Комбинированный анализ



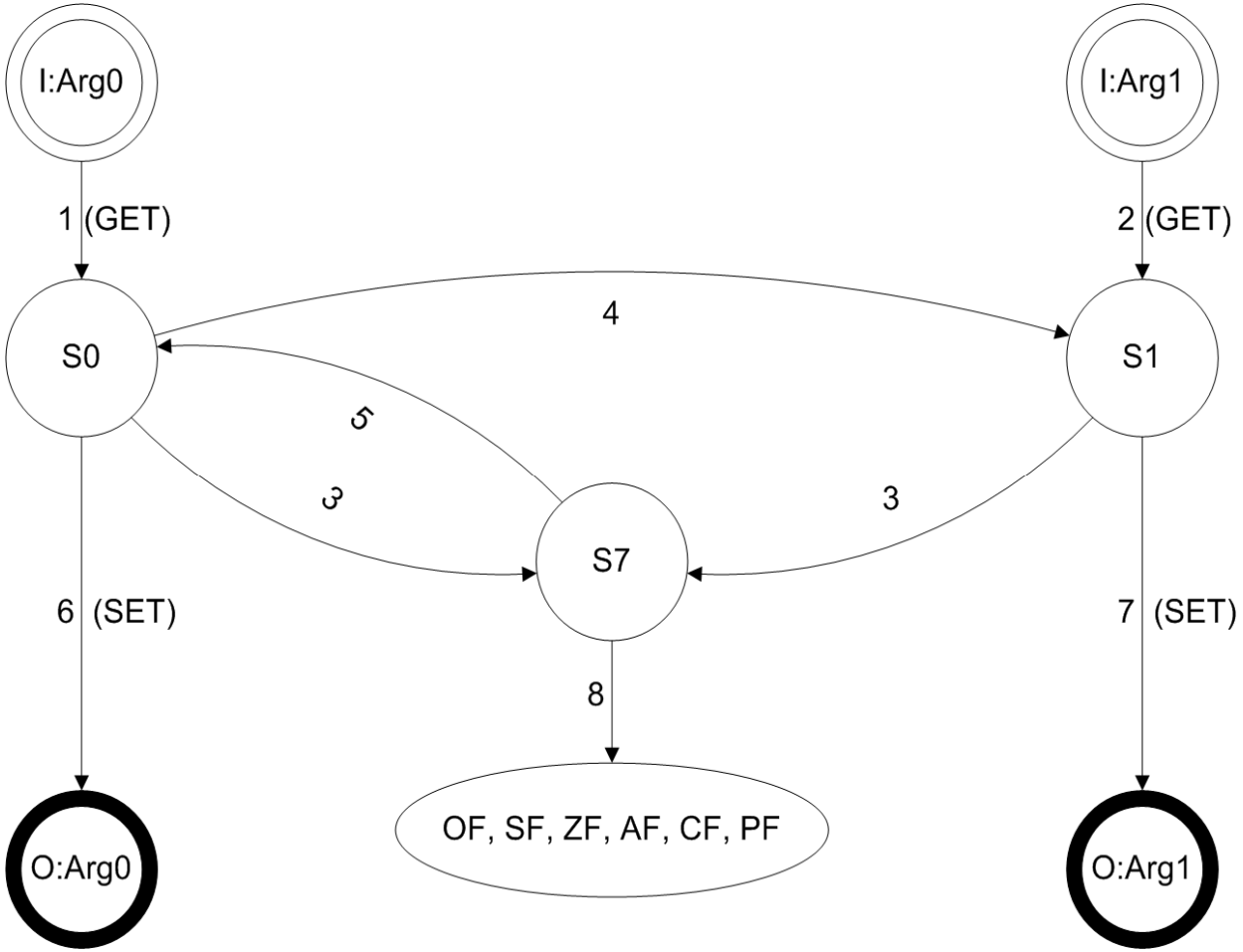
Комбинированный анализ



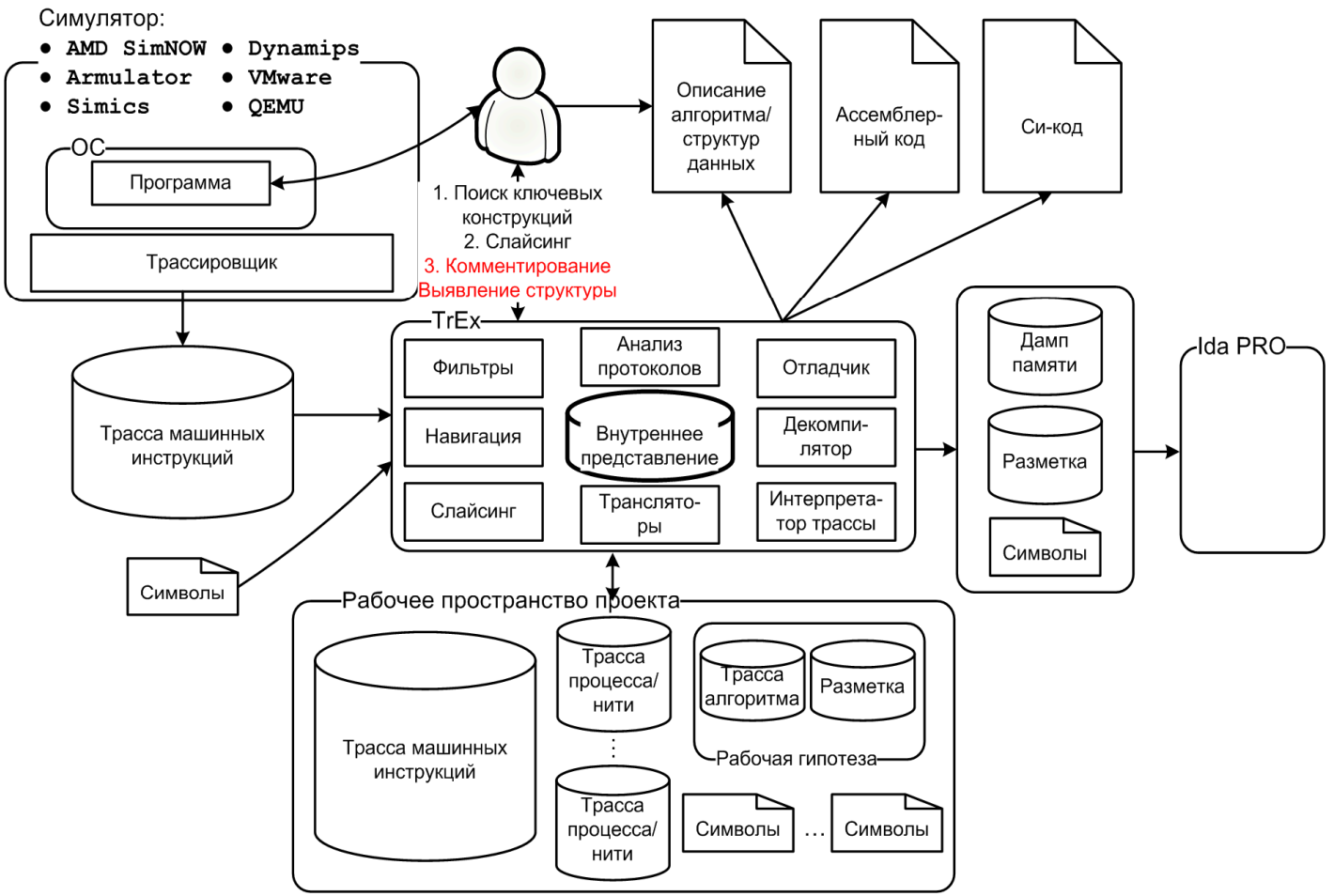
Пример разбора инструкции: XADD

```
DH_BEGIN(XADD, 2, 2, 0)
                /* Забираем оба операнда.  */
                DH_DECOMPOSE_GET    (0);
                DH_DECOMPOSE_GET    (1);

                /* Понадобится временный регистр.  */
                DH_SHADOW           (7, L0              );

                /* Сложение.  */
                DH_UPDATE   ->Outs  (1, S7              )
                            ->Ins   (2, S0              , S1                );

                /* Пересылки.  */
                DH_UPDATE   ->Outs  (1, DH_ED(S1)       )
                            ->Ins   (1, S0              );
                DH_UPDATE   ->Outs  (1, DH_ED(S0)       )
                            ->Ins   (1, S7              );

                /* Заносим в память.  */
                DH_DECOMPOSE_SET    (0);
                DH_DECOMPOSE_SET    (1);

                /* Флаги.  */
    DH_IFPT(F)  DH_UPDATE   ->Outs  (6, DH_REG(OF)      , DH_REG(SF)        , DH_REG(ZF)    ,
                                        DH_REG(AF)      , DH_REG(CF)        , DH_REG(PF)    )
                            ->Ins   (1, S7              );
DH_END
```

Пример разбора инструкции: XADD



Комбинированный анализ



Возможности данного подхода

- Восстановление содержимого буфера с навигацией по шагам трассы, на которых шло заполнение;
- Прямое и обратное выделение алгоритма (прямое выделение алгоритма – по заданным входным ячейкам выделить шаги трассы, участвующие в их обработке; обратное выделение алгоритма – по заданным выходным ячейкам выделить шаги трассы, участвующие в их формировании);
- Генерация простого листинга по трассе;
- Генерация частично работоспособного листинга – расстановка меток и частичная замена адресов метками;
- Отображение графа связи по данным.

Открытые вопросы

- Развитие восстановления формата данных с выходом на модель программы.
- Привлечение методов эмуляции и статического анализа для анализа незадействованных при снятии трассы ветвей алгоритма.
- Разработка внутреннего представления, допускающего эффективную бинарную трансляцию.
- Развитие средств получения трасс.
- Распараллеливание алгоритмов анализа.