Средства симуляции ЦП и ОС и изучение поведения программ

Лекция №5

Державин Андрей
Шурыгин Антон

➢ **Квиз**

- Двоичная трансляция
- Оптимизации
- Прямое исполнение

- Квиз
- **Двоичная трансляция**
- Оптимизации
- Прямое исполнение

Двоичная трансляция

- Какие преимущества и недостатки интерпретирующего симулятора?

Двоичная трансляция

- ➢ Какие преимущества и недостатки интерпретирующего симулятора?
- ➕ Простота реализации
- ➕ Высокая скорость модификации
- ➖ Низкая скорость работы (даже с оптимизациями)

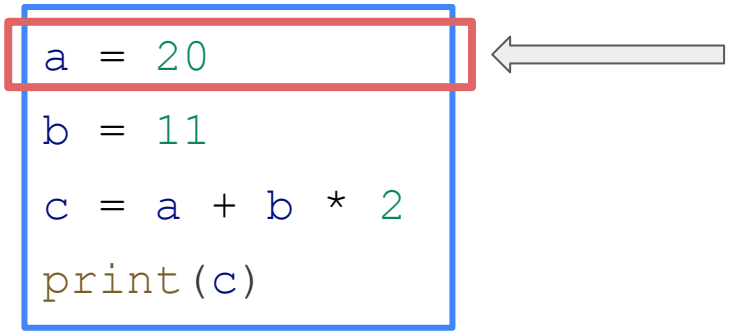
Двоичная трансляция

- Интерпретатор - простой, но медленный

➢ Как мы можем ускорить работу симулятора?

Двоичная трансляция

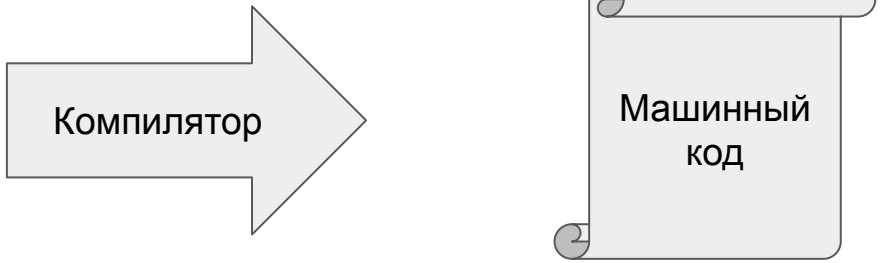
- Интерпретатор - простой, но медленный
- ➢ Как мы можем ускорить работу симулятора?
- Вспомним аналогию с языками программирования



Двоичная трансляция

- Интерпретатор - простой, но медленный
- ➢ Как мы можем ускорить работу симулятора?
- Вспомним аналогию с языками программирования

```
a = 20
b = 11
c = a + b * 2
print(c)
```

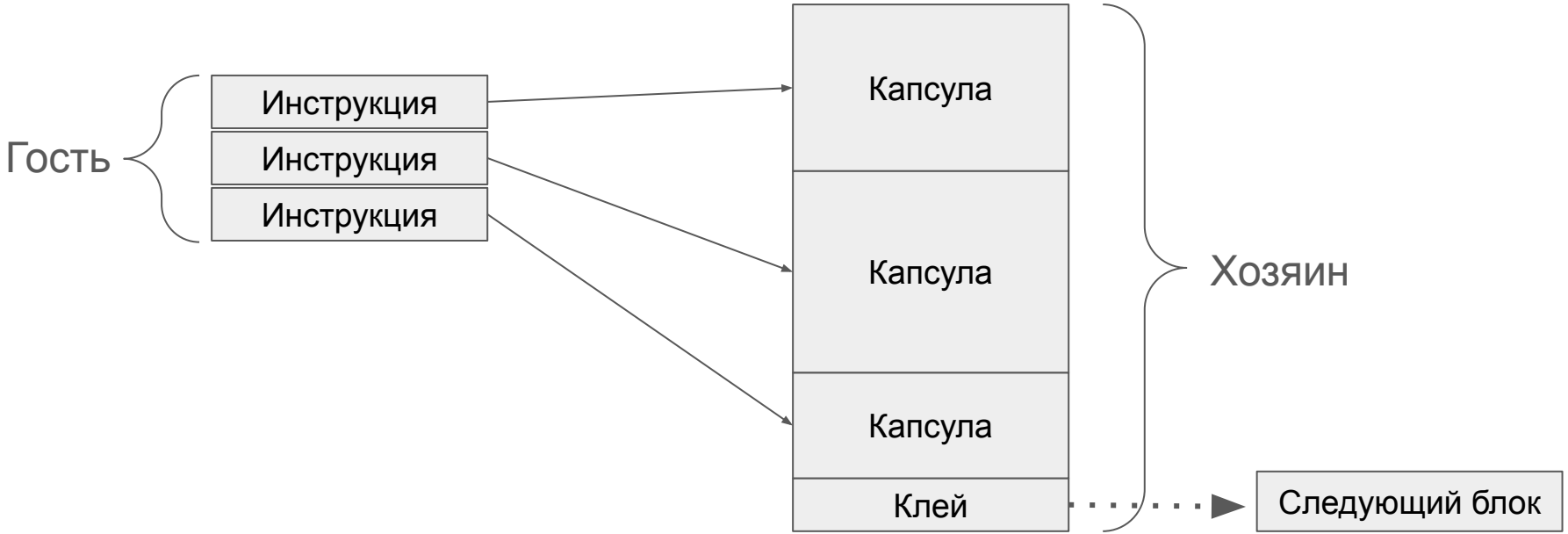


Двоичная трансляция

- Интерпретатор - простой, но медленный
- “Компилируем” блоки кода гостевой архитектуры!

Двоичная трансляция

- Интерпретатор - простой, но медленный
- “Компилируем” блоки кода гостевой архитектуры!



Двоичная трансляция. Капсулы

- Как могут выглядеть капсулы?

Двоичная трансляция. Капсулы

- Рассмотрим капсулу для инструкции add (rv32i)

```
add x1, x2, x3
```

Двоичная трансляция. Капсулы

- Рассмотрим капсулу для инструкции add (rv32i)

```
add x1, x2, x3
```

```
mov edx,DWORD PTR [rdi+0x8]
add edx,DWORD PTR [rdi+0xC]
mov DWORD PTR [rdi+0x4],edx
```

Двоичная трансляция. Капсулы

- Рассмотрим капсулу для инструкции add (rv32i)
- ➢ Регистр rdi хранит адрес начала класса Cpu (регистровый файл)

```
add x1, x2, x3
```

```
mov edx,DWORD PTR [rdi+0x8]
add edx,DWORD PTR [rdi+0xC]
mov DWORD PTR [rdi+0x4],edx
```

Двоичная трансляция. Капсулы

- Рассмотрим капсулу для инструкции add (rv32i)
- Регистр rdi хранит адрес начала класса Cpu (регистровый файл)

➢ Можно ли как-то обобщить капсулу?

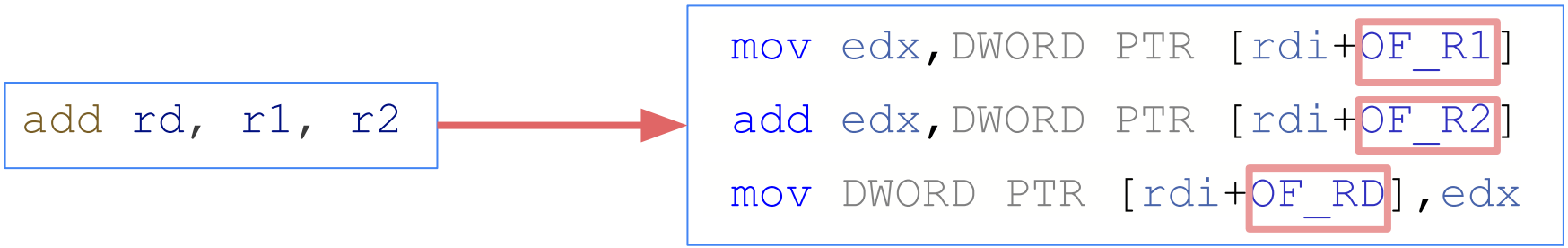
```
add x1, x2, x3
```

```
mov edx,DWORD PTR [rdi+0x8]
add edx,DWORD PTR [rdi+0xC]
mov DWORD PTR [rdi+0x4],edx
```

Двоичная трансляция. Капсулы

- Рассмотрим капсулу для инструкции add (rv32i)
- Регистр rdi хранит адрес начала класса Cpu (регистровый файл)

➢ Храним “шаблон” капсулы для каждой инструкции



Двоичная трансляция

- Разобрались с базовыми понятиями, тут же возникают вопросы

Двоичная трансляция

- Разобрались с базовыми понятиями, тут же возникают вопросы
  - Какая связь во времени у фаз трансляции и симуляции?
  - Как определим единицу бинарной трансляции?

Двоичная трансляция

- Разобрались с базовыми понятиями, тут же возникают вопросы
  - Как определим единицу бинарной трансляции?
  - Какая связь во времени у фаз трансляции и симуляции?

`static`

`dynamic`

Двоичная трансляция

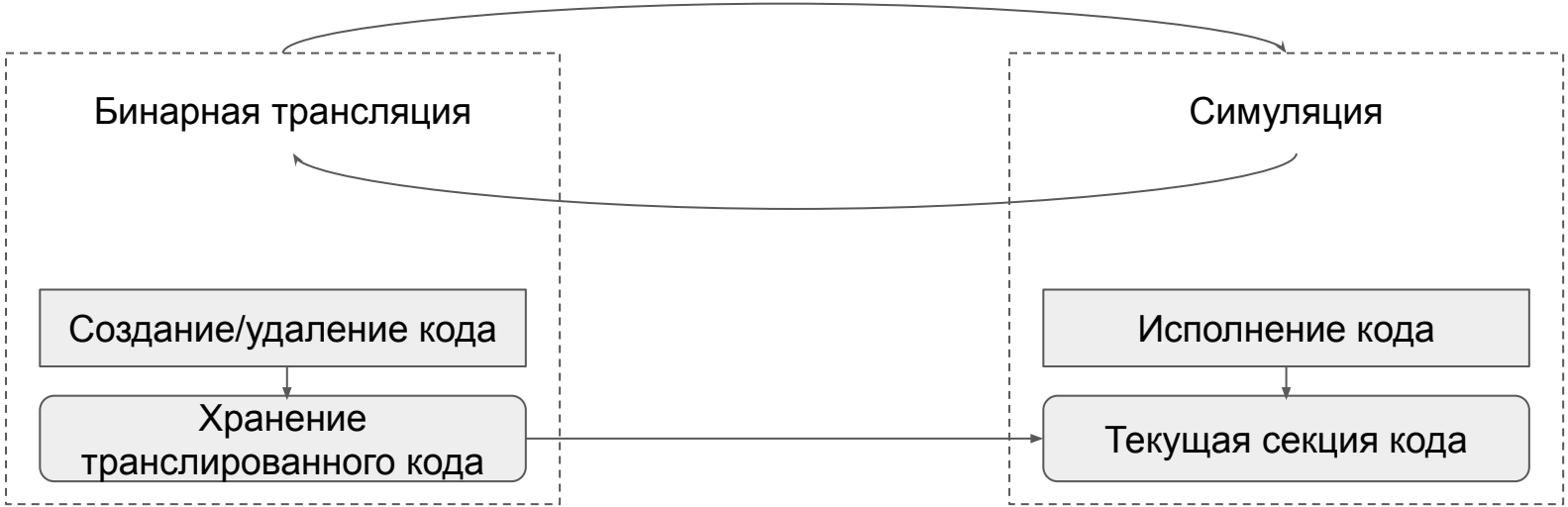
- Статическая бинарная трансляция
- Динамическая бинарная трансляция

Двоичная трансляция

- ➢ Статическая бинарная трансляция – совсем не популярная технология в мире симуляции.
  - ➖ Неэффективное использование ресурсов
  - ➖ Работа кода, изменяющимся в процессе исполнения, некорректна
  - ➕ Статически преобразованный образ можно запускать неограниченное число раз
- ● Динамическая бинарная трансляция

Двоичная трансляция

- Статическая бинарная трансляция
- ➢ Динамическая бинарная трансляция – традиционный подход моделирования гостевой системы.



Двоичная трансляция

- Возвращаемся к упомянутым вопросам
  - Какая связь во времени у фаз трансляции и симуляции?
  - Как определим единицу бинарной трансляции?

Двоичная трансляция

- Возвращаемся к упомянутым вопросам
  - ➢ Какая связь во времени у фаз трансляции и симуляции?
    - Проблема **обнаружения кода**
  - ○ Как определим единицу бинарной трансляции?

Двоичная трансляция

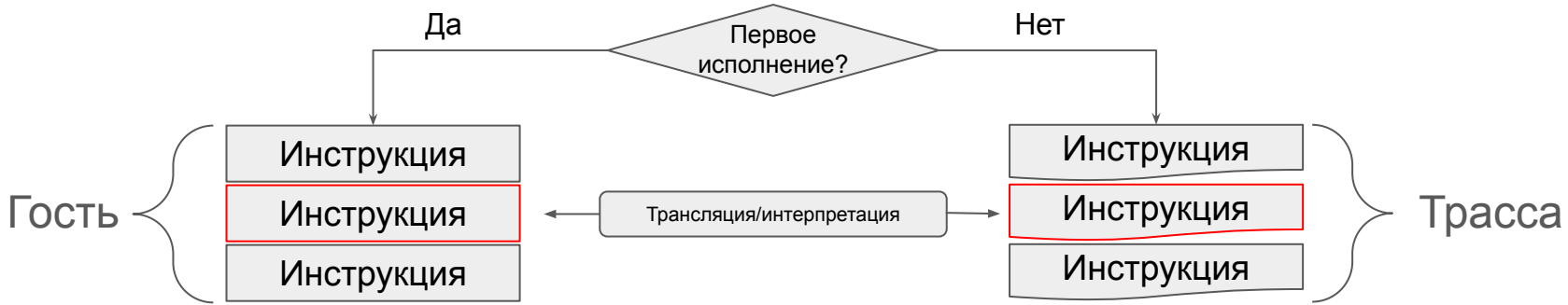
- Возвращаемся к упомянутым вопросам
  - ➢ Какая связь во времени у фаз трансляции и симуляции?
    - Проблема **обнаружения кода**
      1. Регион памяти подвергается трансляции, если вероятность его исполнения ненулевая
      2. Хранить все допустимые точки входа в транслированный блок исполнения
  - ○ Как определим единицу бинарной трансляции?

Двоичная трансляция

- Возвращаемся к упомянутым вопросам
  - ✓ Какая связь во времени у фаз трансляции и симуляции?
  - ➢ Как определим единицу бинарной трансляции?
    - Единица трансляции
      1. Трасса исполнения
      2. Гостевая страница

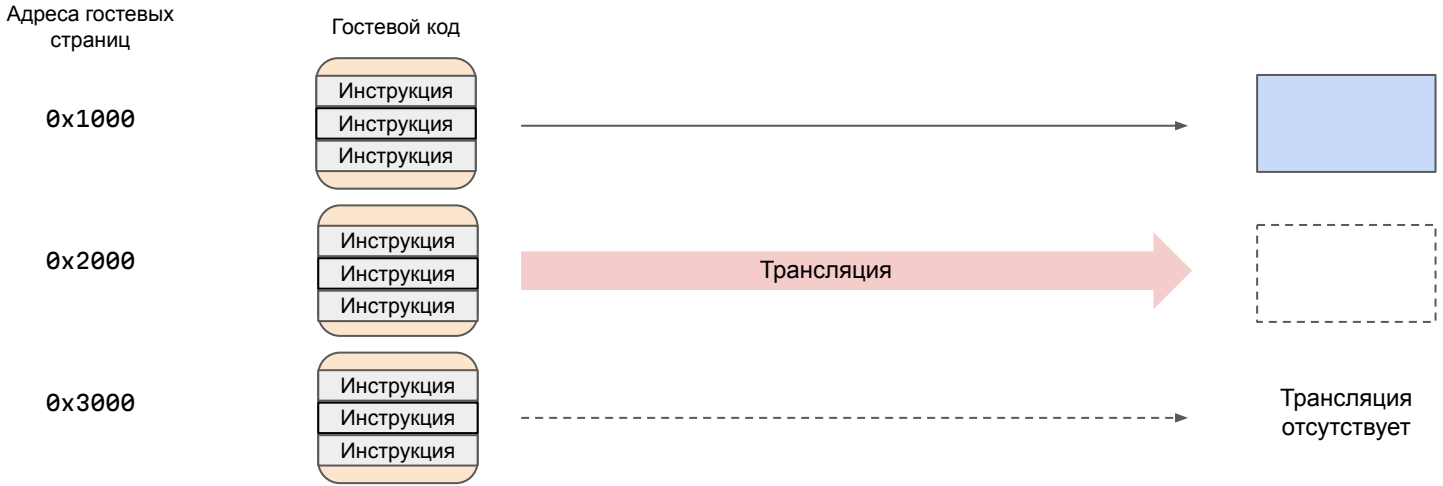
Двоичная трансляция

- Возвращаемся к упомянутым вопросам
  - ✓ Какая связь во времени у фаз трансляции и симуляции?
  - ○ Как определим единицу бинарной трансляции?
    - ■ Единица трансляции
      - ➢ Трасса исполнения
      1. Гостевая страница



Двоичная трансляция

- Возвращаемся к упомянутым вопросам
  - ✓ Какая связь во времени у фаз трансляции и симуляции?
  - ○ Как определим единицу бинарной трансляции?
    - ■ Единица трансляции
      1. Трасса исполнения
      - ➢ Гостевая страница



Понятие SMC

- Обычно исполняемый код и обрабатываемые данные находятся в одной физической памяти
- Создание программ, которые в процессе работы изменяют код других программ, в т.ч. свой собственный
- Такое явление получило название самомодифицирующийся код (англ. self-modifying code, SMC)

Проблема SMC

- Гостевой код изменяется при исполнении SMC программы.
- ➢ Проблема: возникает вероятность, что уже оттранслированные блоки кода провисли и перестали соответствовать содержимому памяти модели.

Проблема SMC

- Гостевой код изменяется при исполнении SMC программы.

✓ Проблема: возникает вероятность, что уже оттранслированные блоки кода провисли и перестали соответствовать содержимому памяти модели.

- Решение – проверка всех записей в память для дальнейшего сброса состояния модели и повторной трансляции затронутых блоков.

➢ Снова проблема – долгий процесс бинарной трансляции одного блока:

  - Скорость работы симулятора снижается
  - Простой интерпретатор оказывается более производительным

- Квиз
- Двоичная трансляция
- **Оптимизации**
- Прямое исполнение

Оптимизирующая трансляция

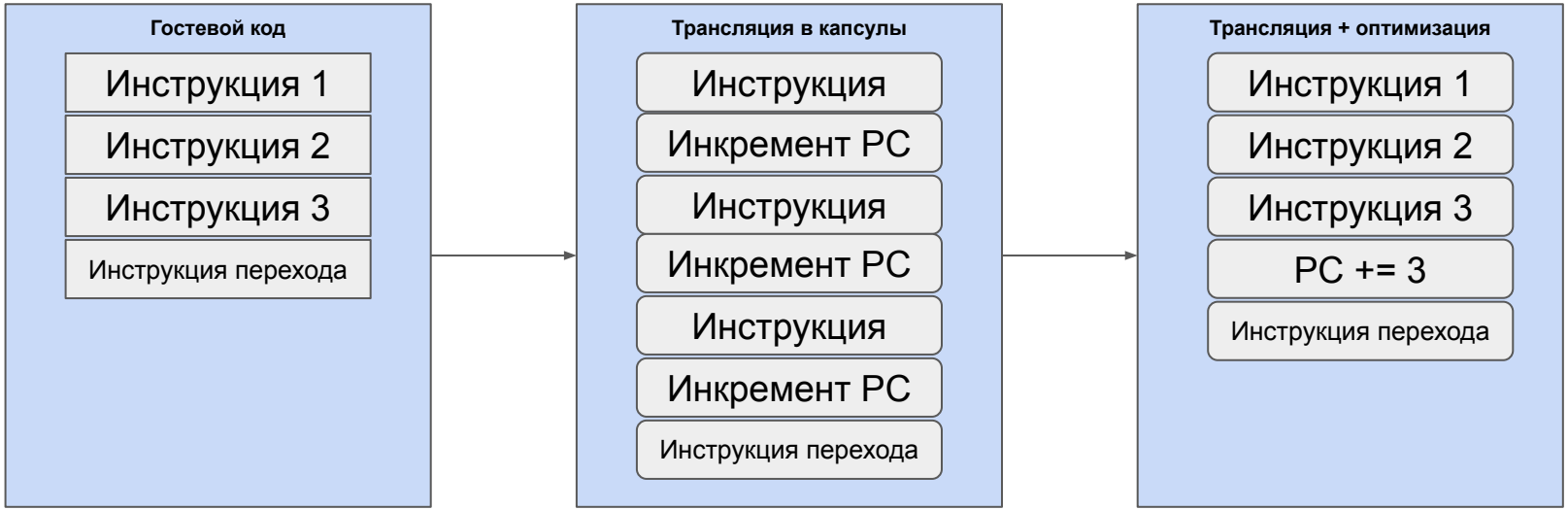
- Бинарная трансляция привносит не только сложности и проблемы в работе модели
- Ведь у нас появляется механизм преобразовать блок трансляции так, чтобы он работал быстрее

Оптимизирующая трансляция

- Бинарная трансляция привносит не только сложности и проблемы в работе модели
- Ведь у нас появляется механизм преобразовать блок трансляции так, чтобы он работал быстрее
- Какие компиляторные оптимизации вы знаете?

Оптимизирующая трансляция

- Рассмотрим пример часто используемой оптимизации (Helmstetter et al., 2011)



Оптимизирующая трансляция

- Компиляторные оптимизации, применяемые и при двоичной трансляции:
  - Удаление мертвого кода (англ. dead code elimination)
  - Удаление общих подвыражений (англ. common subexpression elimination)
  - Свертка констант (англ. constant folding) и дублирование констант (англ. constant propagation)
  - Peephole оптимизация

- Квиз
- Двоичная трансляция
- Оптимизации
- ➢ **Прямое исполнение**

А что если host = guest?

- Важный случай бинарной трансляции – совпадение гостевой и хозяйской архитектуры
- Возникает желание скопировать гостевой код как хозяйский или даже *исполнить его напрямую*
- Подобный режим симуляции называется прямое исполнение (англ. direct execution, DEX)

Прямое исполнение

- Хозяйская архитектура совпадает с гостевой
- В любом случае необходимо обеспечить изоляцию исполнения кода гостевых приложений от кода симулятора
- Примеры операций, нарушающих условия изолированности:
  - Доступ к памяти и периферии
  - Архитектурное состояние
  - Привилегированные инструкции

Прямое исполнение

- Какие у вас есть идеи, как можно обойти все эти уязвимости?

Прямое исполнение

- Предпросмотр кода – процесс инспектирования гостевого кода на предмет инструкций, запрещенных к ***прямому исполнению*** и/или требующих выполнения дополнительных действий, связанных с работой симулятора
- В результате предпросмотра проблемные инструкции мы можем “обезвредить”:
    - Заплатка (англ. patch)
    - Заглушка (англ. stub)

Прямое исполнение

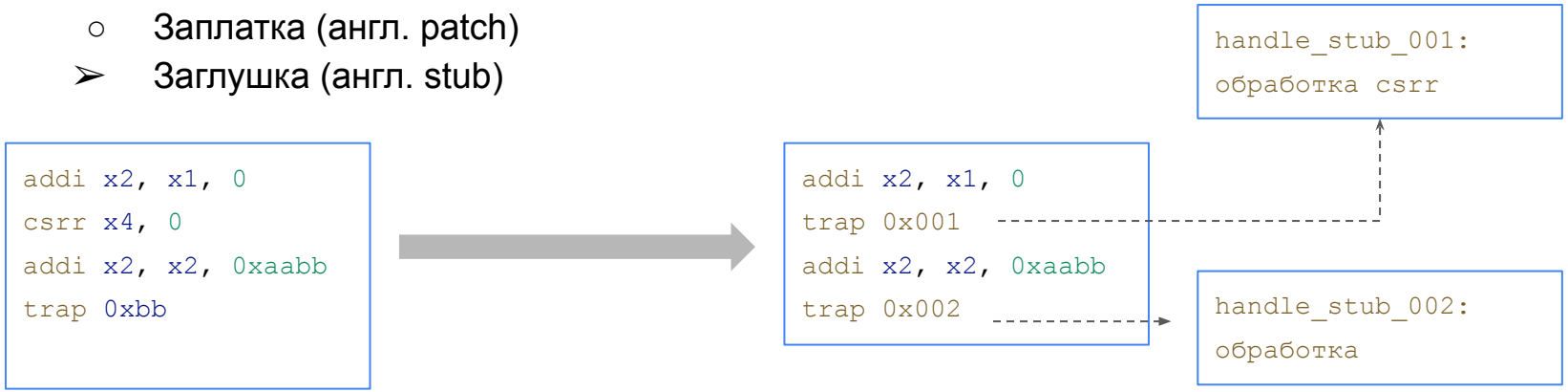
- Предпросмотр кода – процесс инспектирования гостевого кода на предмет инструкций, запрещенных к ***прямому исполнению*** и/или требующих выполнения дополнительных действий, связанных с работой симулятора
- В результате предпросмотра проблемные инструкции мы можем “обезвредить”:
  - ➢ Заплатка (англ. patch)
  - ○ Заглушка (англ. stub)

```
addi x2, x1, 0
addi x2, x2, 0xaabb
lw x3, 0x100(x0)
sw x2, 0x200(x0)
jal x1, 0x3040
```

```
addi x2, x1
addi x2, x2
lui x4, 0xf00
lw x3, 0x100(x4)
lui x5, 0xf00
sw x2, 0x200(x5)
lui x6, 0xf003
jalr x1, 0x040(x6)
```

Прямое исполнение

- Предпросмотр кода – процесс инспектирования гостевого кода на предмет инструкций, запрещенных к ***прямому исполнению*** и/или требующих выполнения дополнительных действий, связанных с работой симулятора
- В результате предпросмотра проблемные инструкции мы можем “обезвредить”:
  - Заплатка (англ. patch)
  - ➢ Заглушка (англ. stub)



Литература

1. Richard L. Sites, Anton Chernoff, Matthew B. Kirk, Maurice P. Marks, and Scott G. Robinson. Binary translation.
2. Claude Helmstetter, Vania Joloboff, Zhou Xinlei, and Gao Xiaopeng. Fast instruction set simulation using LLVM-based dynamic translation.
3. Nigel Topham and Daniel Jones. High speed CPU simulation using JIT binary translation.