Двоичная трансляция

Курс «Программное моделирование вычислительных систем»

Григорий Речистов
grigory.rechistov@phystech.edu

23 марта 2015 г.

На (поза)прошлой лекции

- Интерпретаторы — медленная шутка
- Рассмотренные улучшения основаны на повторном использовании уже полученных результатов
- Существуют устоявшиеся идиомы для представления моделируемого архитектурного состояния

Вопросы

- Сколько бит в машинном слове?

Вопросы

- Сколько бит в машинном слове?
- Что лучше — MMIO или PIO?

Вопросы

- Сколько бит в машинном слове?
- Что лучше — MMIO или PIO?
- Может ли архитектура быть и не little, и не big-endian?

Что удалось соптимизировать в интерпретаторе

- **Fetch** ← оптимизировано
- **Decode** ← оптимизировано
- Execute
- Writeback
- Advance PC

Что удалось соптимизировать в интерпретаторе

- Fetch
- Decode
- **Execute** $\rightarrow$ осталось соптимизировать
- **Writeback** $\rightarrow$ осталось соптимизировать
- **Advance PC** $\rightarrow$ осталось соптимизировать

Интерпретация и трансляция в языках высокого уровня

- BASIC, CPython, Shell...
  - Прочитать строку — распознать команды — исполнить
  - Медленно работает, но больше «интерактивности»
- Fortran, C, Pascal...
  - Первый проход: распознавание команд языка и преобразование их в машинный код
  - Второй проход: исполнение машинного кода

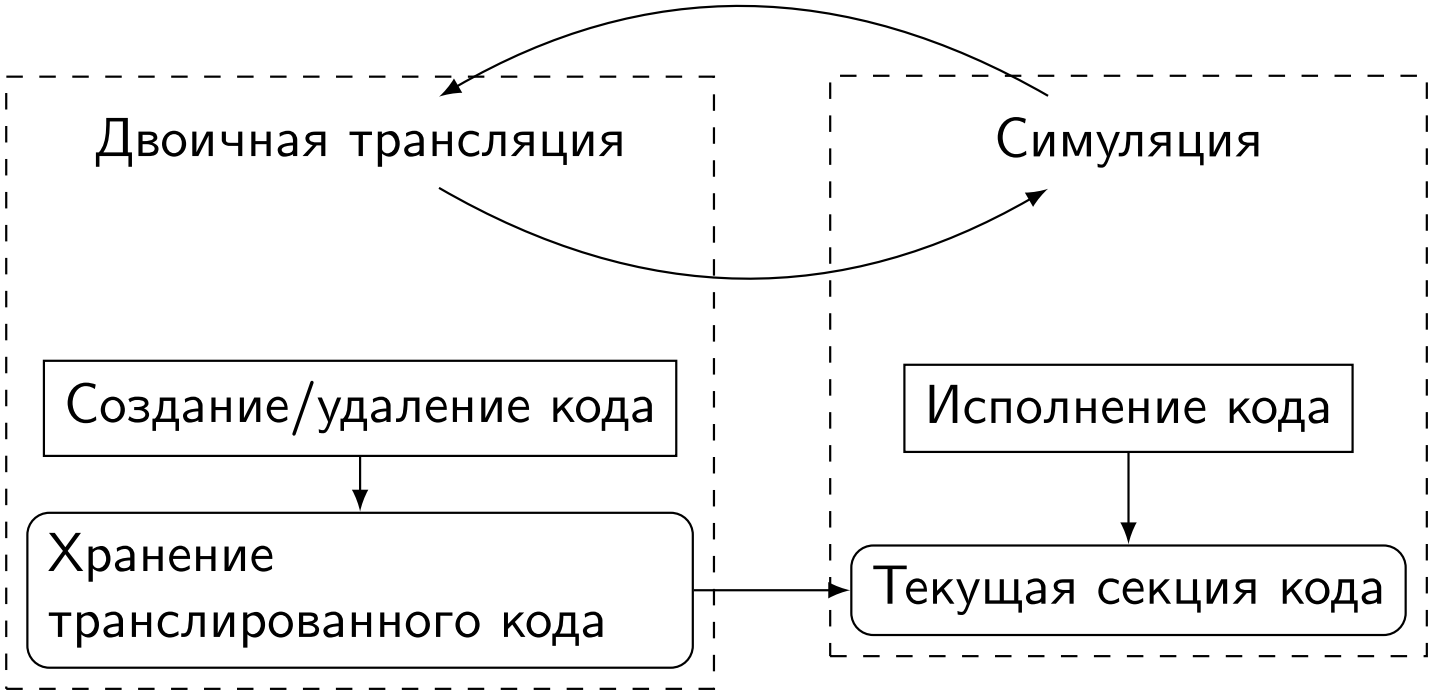
Двоичная трансляция

- Входной язык — гостевой машинный код
- Целевой язык — хозяйский машинный код
- ДТ — перевод кода гостевой программы, записанной в гостевой ISA, в эквивалентный код в терминах хозяйской ISA
- Ради чего: многократное исполнение результатов трансляции

Статическая и динамическая ДТ

- *Статическая* трансляция исполняется заранее, до исполнения первой инструкции
- Результат статической трансляции сохраняется на диске
- *Динамическая* трансляция происходит непосредственно во время симуляции
- Результат динамической трансляции хранится в памяти
- Динамическая трансляция чередуется с исполнением генерированного кода

Фазы динамической ДТ



Алгоритм 1: шаблонная трансляция

Алгоритм 1: шаблонная трансляция

- start_addr — гостевой адрес начала кода
- start_buf — хозяйский буфер

```
translate(start_addr, start_buf) {
    PC = start_addr; bufptr = start_buf;
    while (!enough) {
        instr = fetch(PC);
        (opcode, operands) = decode(instr);
        (template, length) = templates[opcode];
        memcpy(bufptr, template, length);
        patch_operands(bufptr, operands);
        PC += instr_length;
        bufptr += length;
    }
    memcpy(bufptr, glue_capsule, glue_length);
}
```

Алгоритм 1: исполнение

```
execute(start_buf) {
    load_simulated_state();
    goto start_buf;
}
```

Алгоритм 1: исполнение

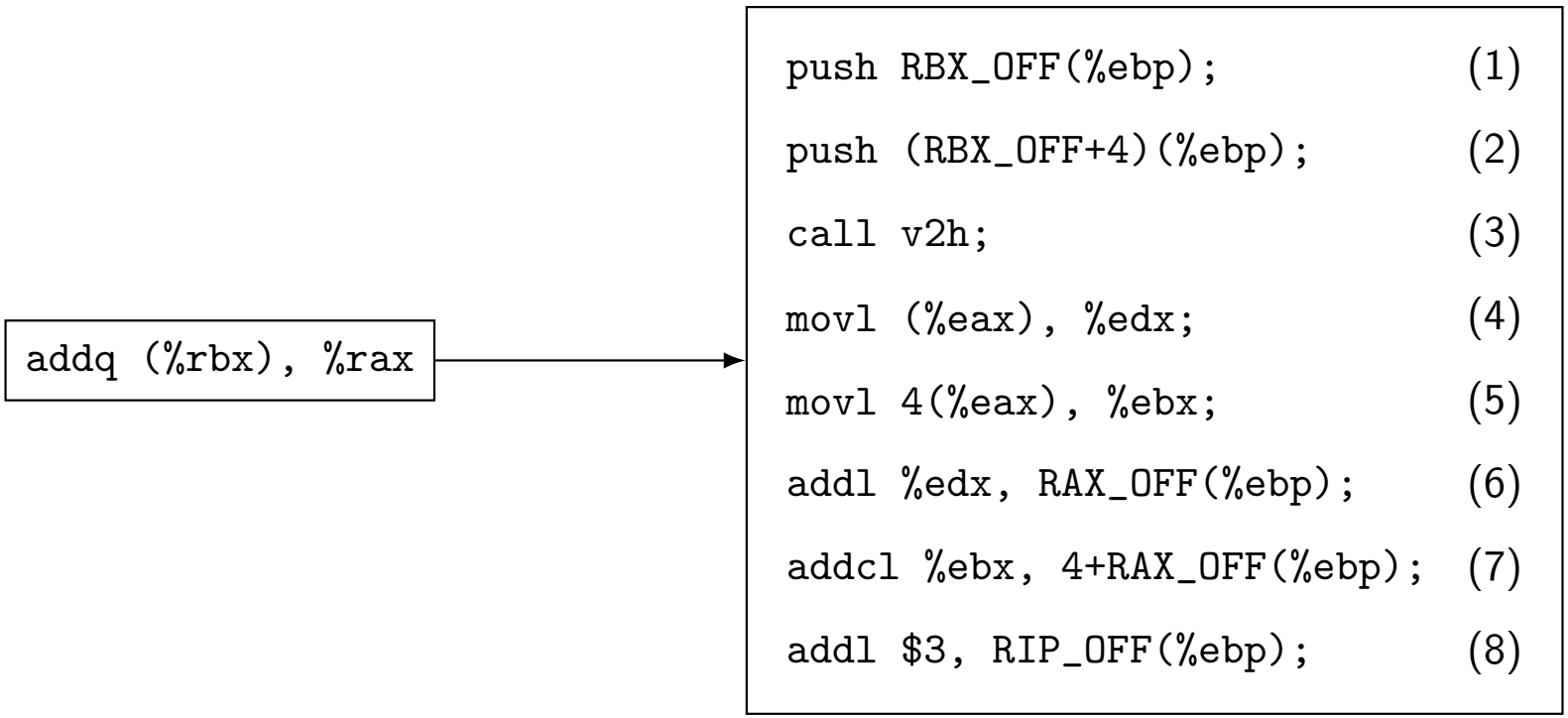
```
execute(start_buf) {
    load_simulated_state();
    goto start_buf;
}
```

или

```
typedef void (*fblock)(void);
execute(start_buf) {
    load_simulated_state();
    ((fblock)start_buf)();
}
```

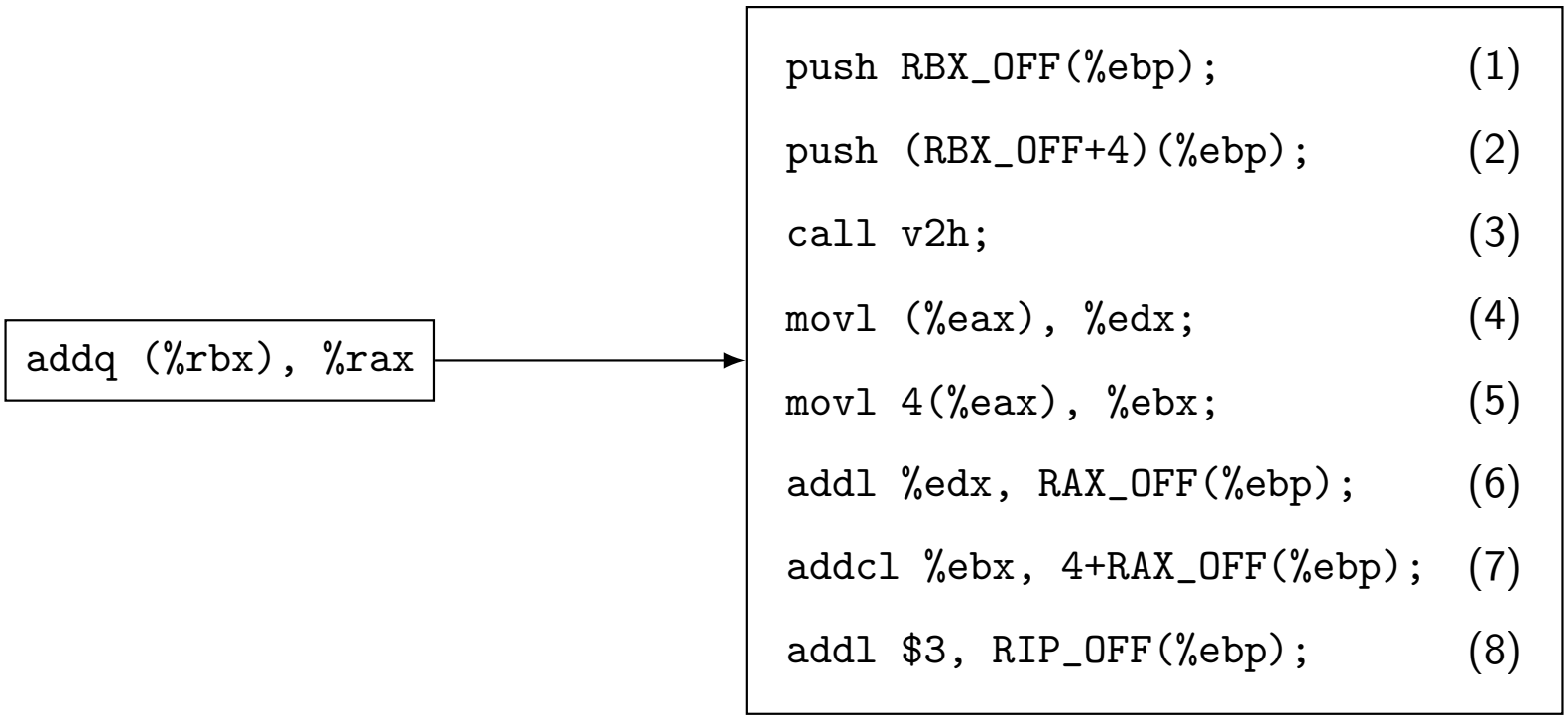
Капсула

Гостевой код, Intel 64 (64 бит) Хозяйский код, IA-32 (32 бит)



Капсула

Гостевой код, Intel 64 (64 бит) Хозяйский код, IA-32 (32 бит)



Вопрос: что из семантики ADDQ забыто в описании капсулы?

Подстановка аргументов в капсулу

Регистры:

```
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 f4 58 c9      vaddps %ymm1,%ymm1,%ymm1
c5 f4 58 cf      vaddps %ymm7,%ymm1,%ymm1
```

Подстановка аргументов в капсулу

Регистры:

```
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 f4 58 c9      vaddps %ymm1,%ymm1,%ymm1
c5 f4 58 cf      vaddps %ymm7,%ymm1,%ymm1
c4 c1 74 58 c8   vaddps %ymm8,%ymm1,%ymm1
c4 c1 74 58 cf   vaddps %ymm15,%ymm1,%ymm1
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 ec 58 d0      vaddps %ymm0,%ymm2,%ymm2
c5 c4 58 f8      vaddps %ymm0,%ymm3,%ymm3
```

Подстановка аргументов в капсулу

Регистры:

```
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 f4 58 c9      vaddps %ymm1,%ymm1,%ymm1
c5 f4 58 cf      vaddps %ymm7,%ymm1,%ymm1
c4 c1 74 58 c8   vaddps %ymm8,%ymm1,%ymm1
c4 c1 74 58 cf   vaddps %ymm15,%ymm1,%ymm1
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 ec 58 d0      vaddps %ymm0,%ymm2,%ymm2
c5 c4 58 f8      vaddps %ymm0,%ymm3,%ymm3
c4 e1 74 58 c8   vaddps %ymm0,%ymm1,%ymm1 # Мнемоника та же!
```

Подстановка аргументов в капсулу

Регистры:

```
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 f4 58 c9      vaddps %ymm1,%ymm1,%ymm1
c5 f4 58 cf      vaddps %ymm7,%ymm1,%ymm1
c4 c1 74 58 c8   vaddps %ymm8,%ymm1,%ymm1
c4 c1 74 58 cf   vaddps %ymm15,%ymm1,%ymm1
c5 f4 58 c8      vaddps %ymm0,%ymm1,%ymm1
c5 ec 58 d0      vaddps %ymm0,%ymm2,%ymm2
c5 c4 58 f8      vaddps %ymm0,%ymm3,%ymm3
c4 e1 74 58 c8   vaddps %ymm0,%ymm1,%ymm1 # Мнемоника та же!
```

Литералы:

```
67 c7 85 00 01 00 00 dd cc bb aa  movl $0xaabbccdd,0x100(%ebp)
```

Алгоритм 2: JIT. Генерация IR

Алгоритм 2: JIT. Стадия симуляции

Оптимизации

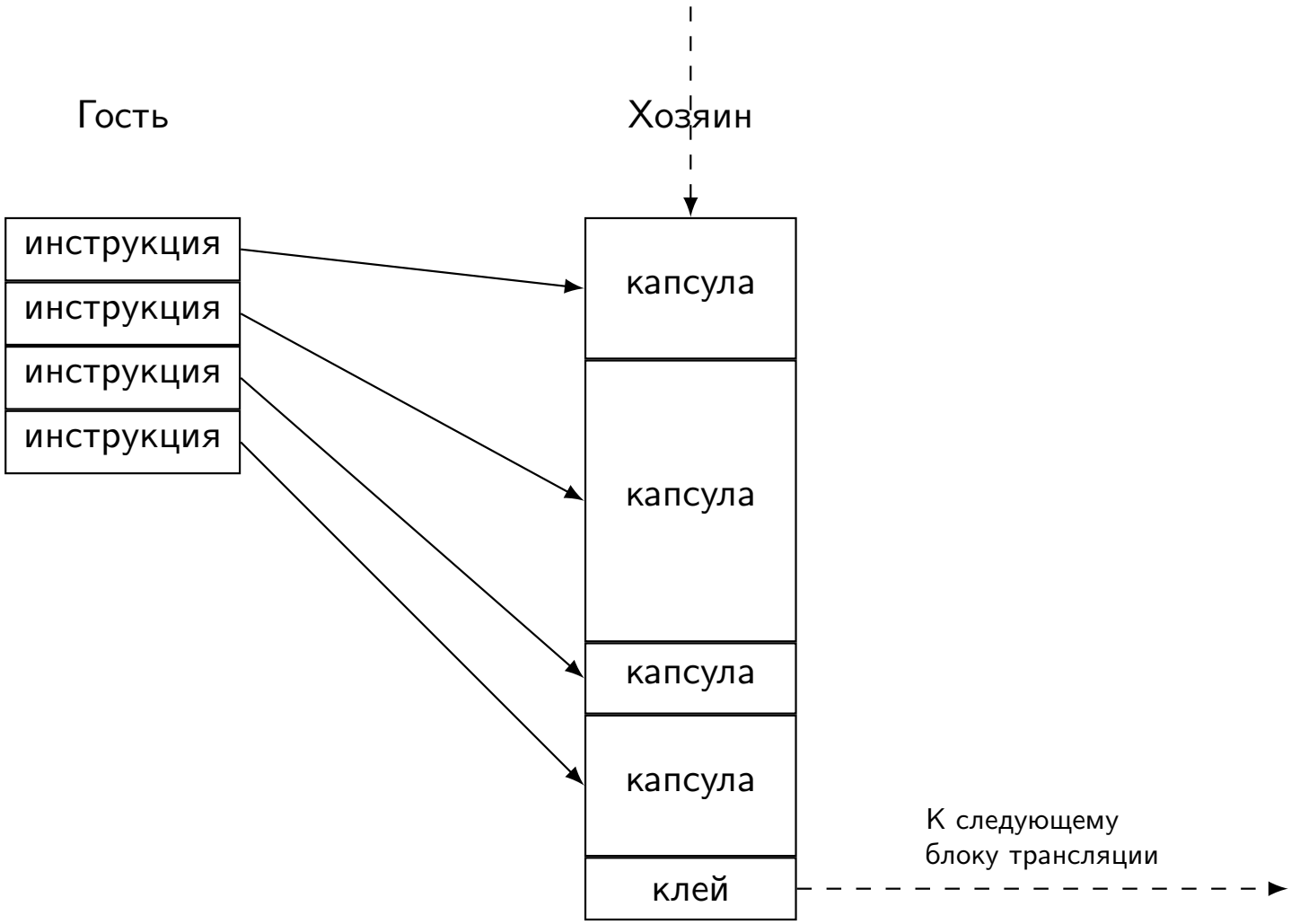
Гостевой код → ДТ → Оптимизация ДТ

```
instr1
instr2
instr3
instr4
instr5
branch
```

```
<instr1>
inc PC_OFF(%r14)
<instr2>
inc PC_OFF(%r14)
<instr3>
inc PC_OFF(%r14)
<instr4>
inc PC_OFF(%r14)
<instr5>
inc PC_OFF(%r14)
<branch>
```

```
<instr1>
<instr2>
<instr3>
<instr4>
<instr5>
add $5, PC_OFF(%r14)
<branch>
```

Связь между блоками трансляции



Почему оптимизации при ДТ затруднительны

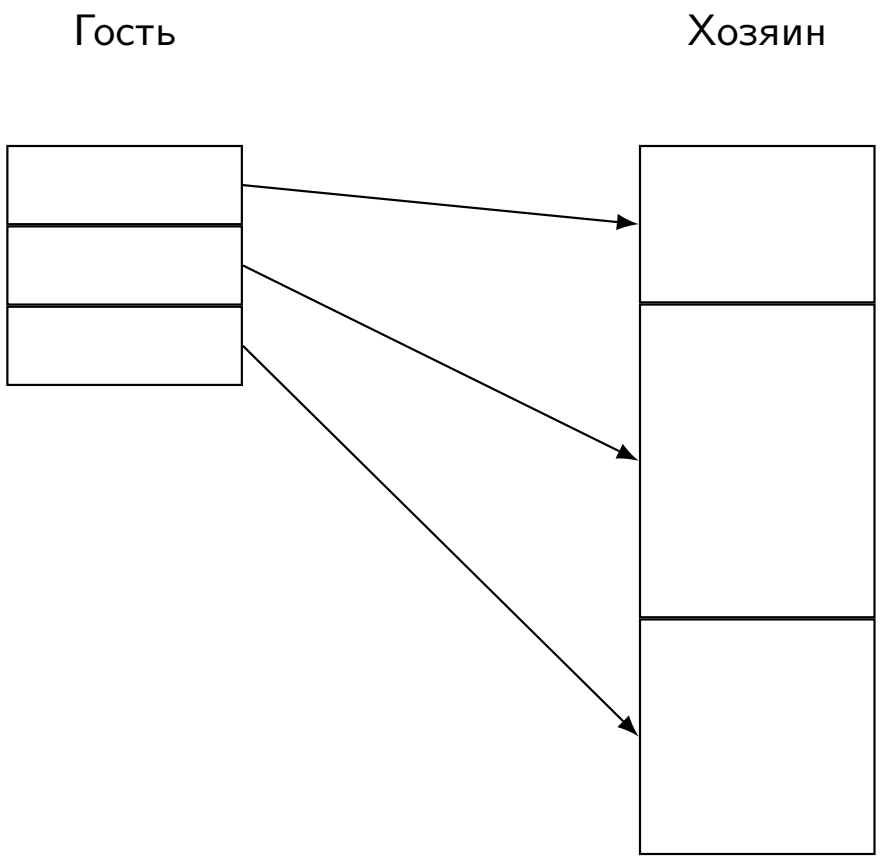
- В отличие от ЯВО, машинный код содержит меньше информации об исходном алгоритме
- Нельзя делать многие предположения, необходимые для компиляторных оптимизаций, без нарушения корректности
- В случае динамической ДТ оптимизации ограничены длительностью их работы

Почему оптимизации при ДТ затруднительны

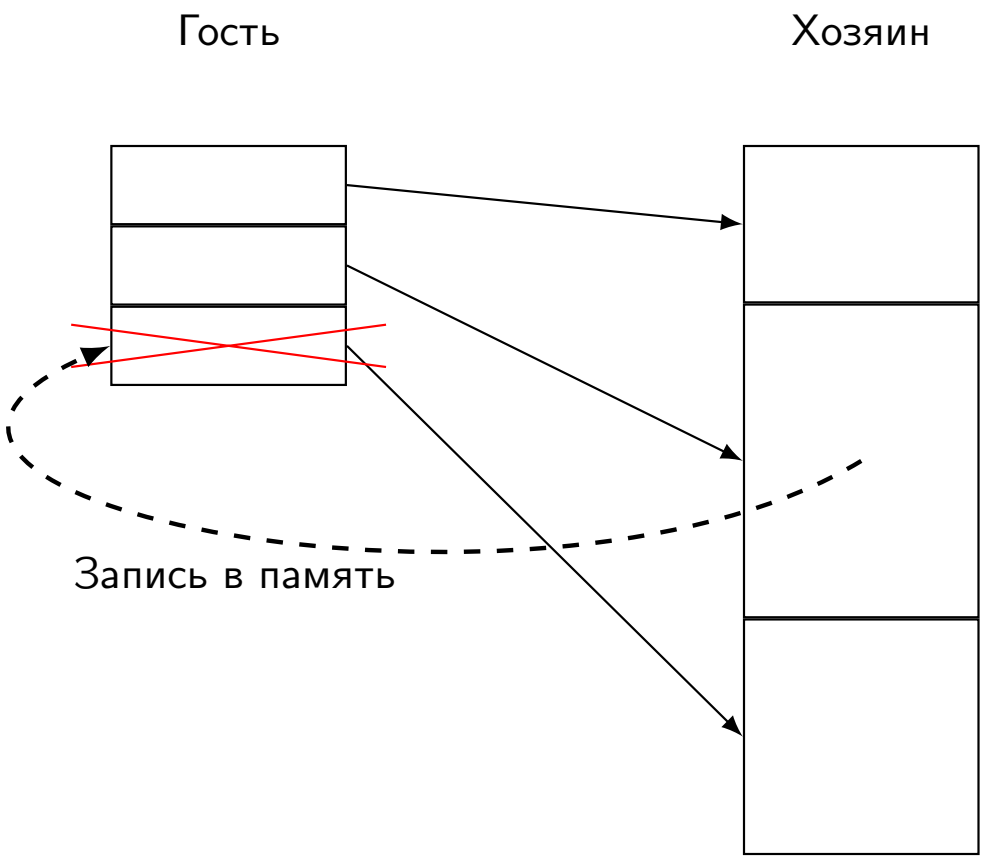
- В отличие от ЯВО, машинный код содержит меньше информации об исходном алгоритме
- Нельзя делать многие предположения, необходимые для компиляторных оптимизаций, без нарушения корректности
- В случае динамической ДТ оптимизации ограничены длительностью их работы

- Адреса переменных — их нет
- Границы процедур — их нет
- Адреса переходов — известна только часть из них

Самомодифицирующийся код (self modifying code)

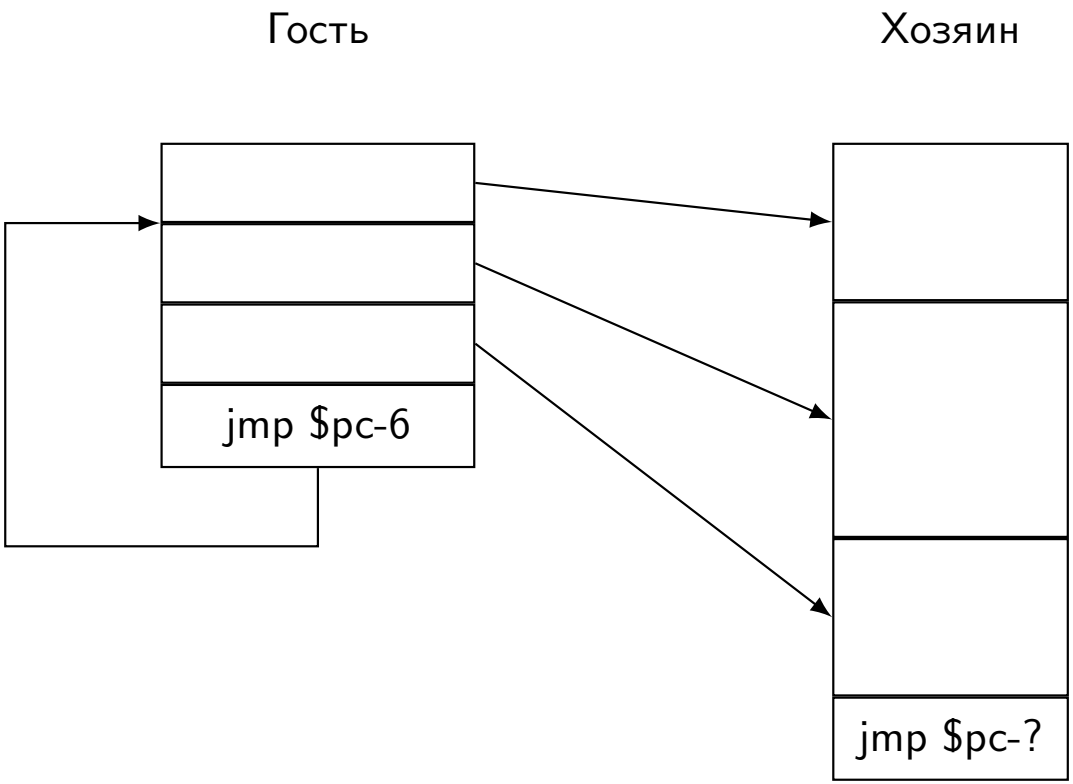


Самомодифицирующийся код (self modifying code)



Обнаружение кода 1

- Найти границы инструкций
- Отличить код от данных



Обнаружение кода 2

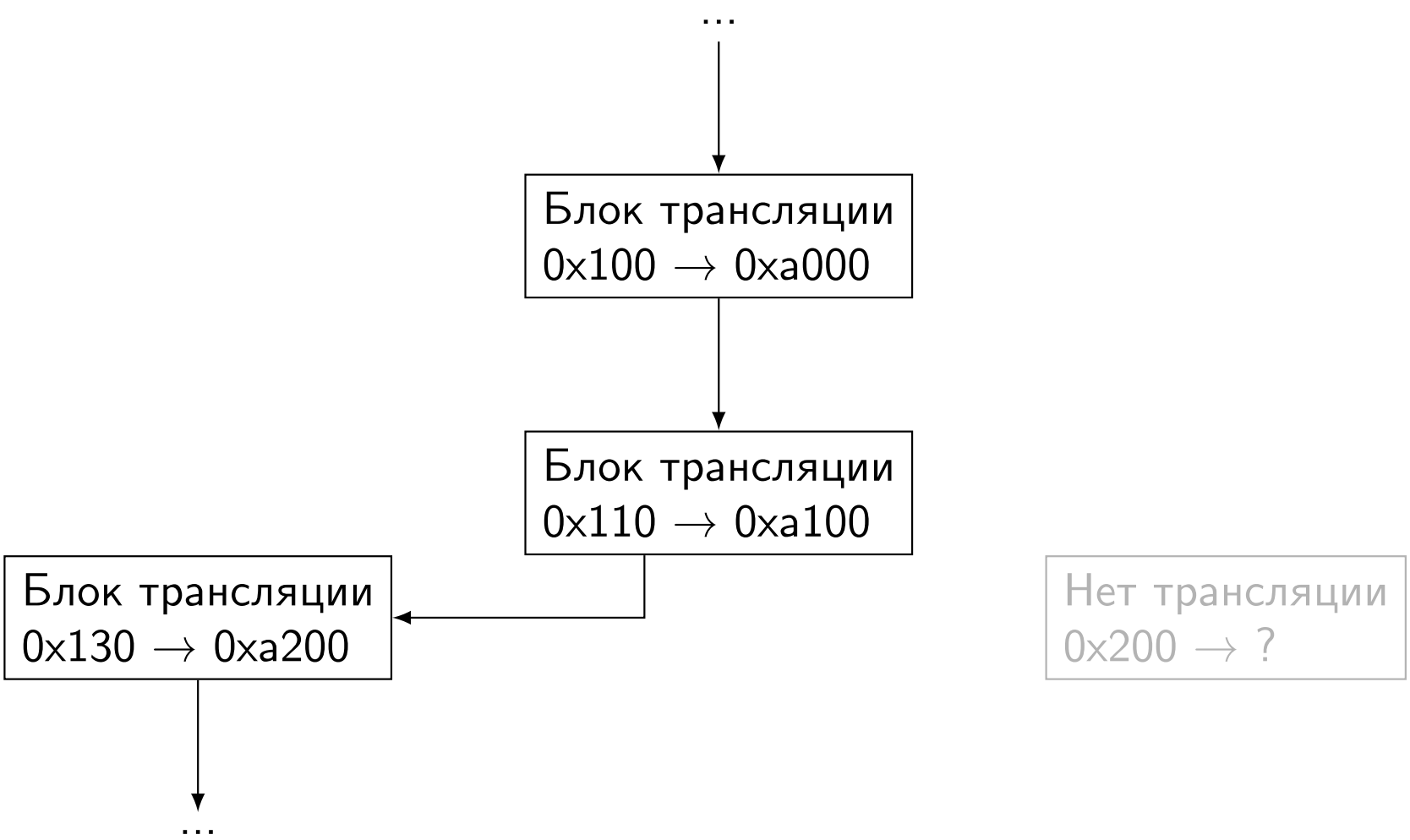
```
vpcmpgtq %xmm1,%xmm2,%xmm3
in $0x90,%al
```

```
0xc4 0xe2 0x69 0x37 0xd9 0xe4 0x90
```

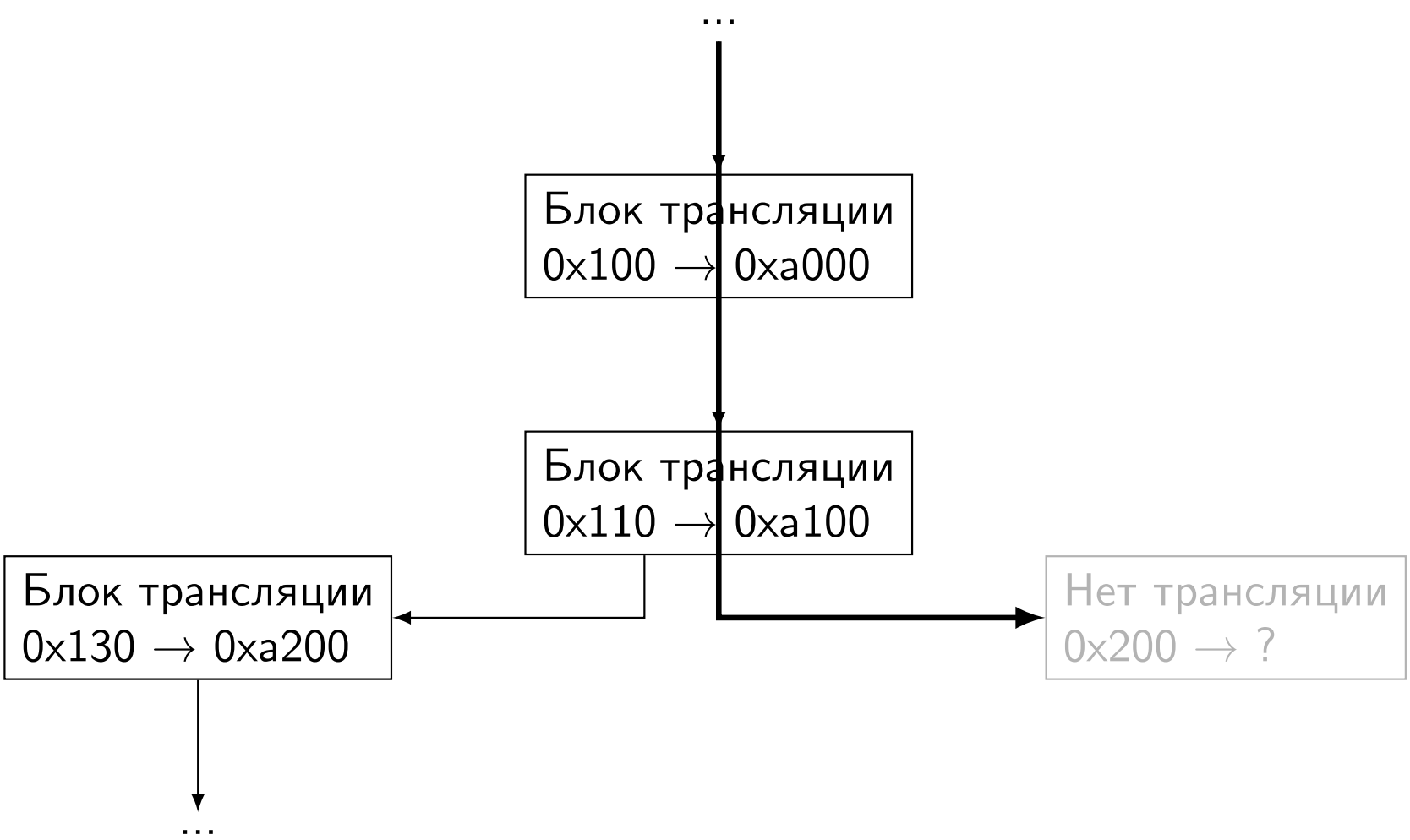
```
loop 6b <.text+0x6b>
aaa
ftst
nop
```

По этой причине статическое пре-декодирование не всегда возможно

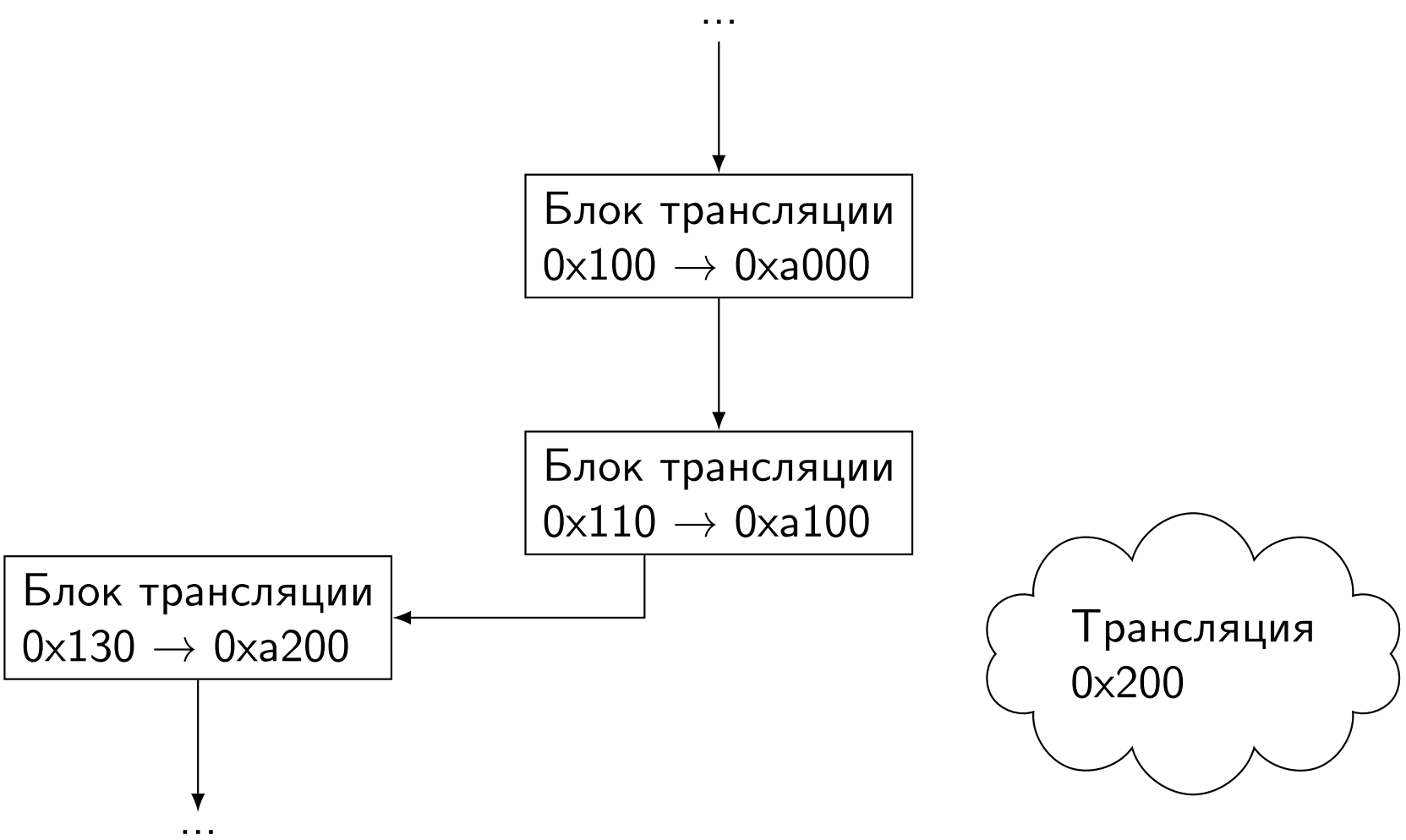
Опережающая трансляция



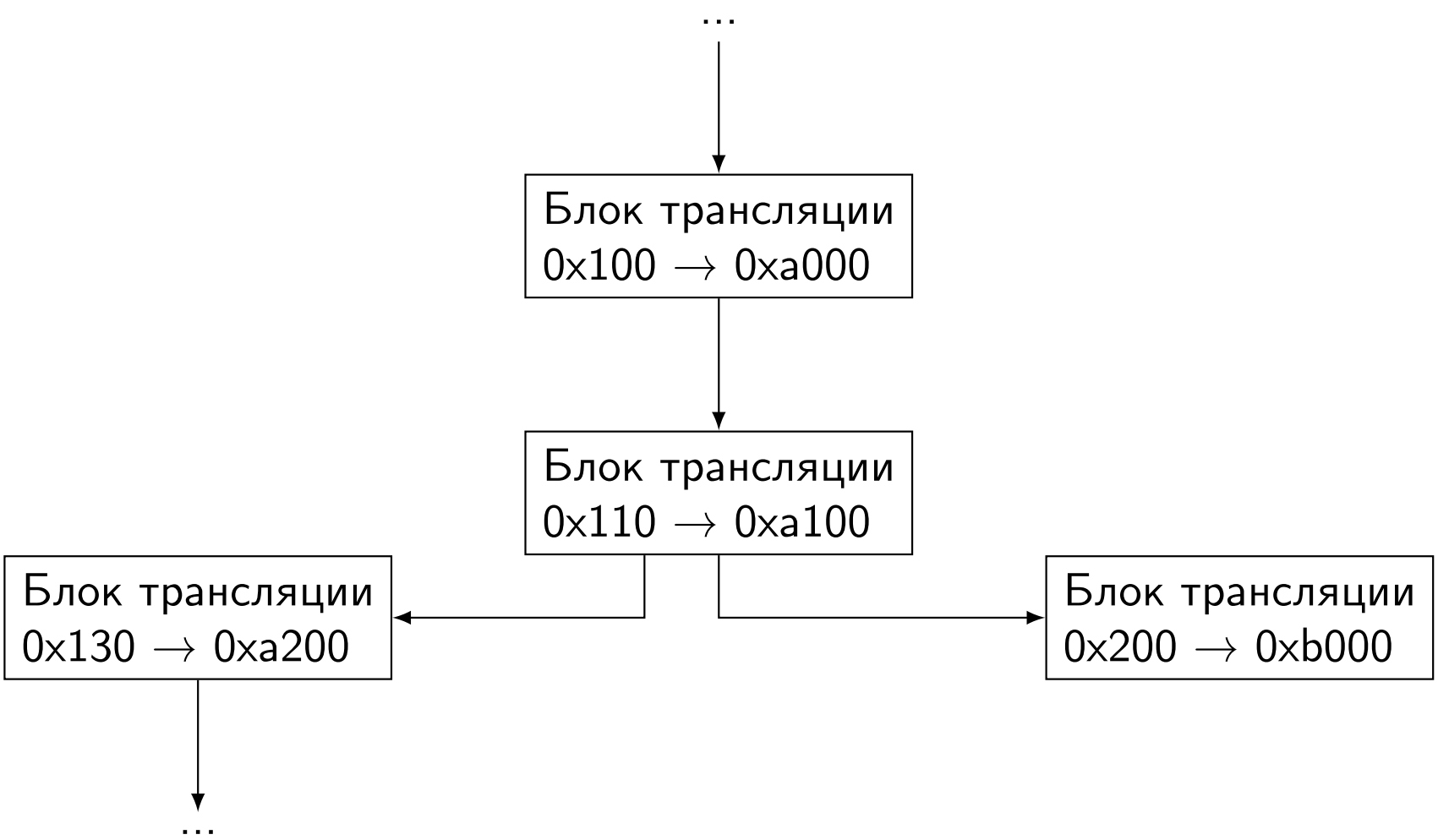
Опережающая трансляция



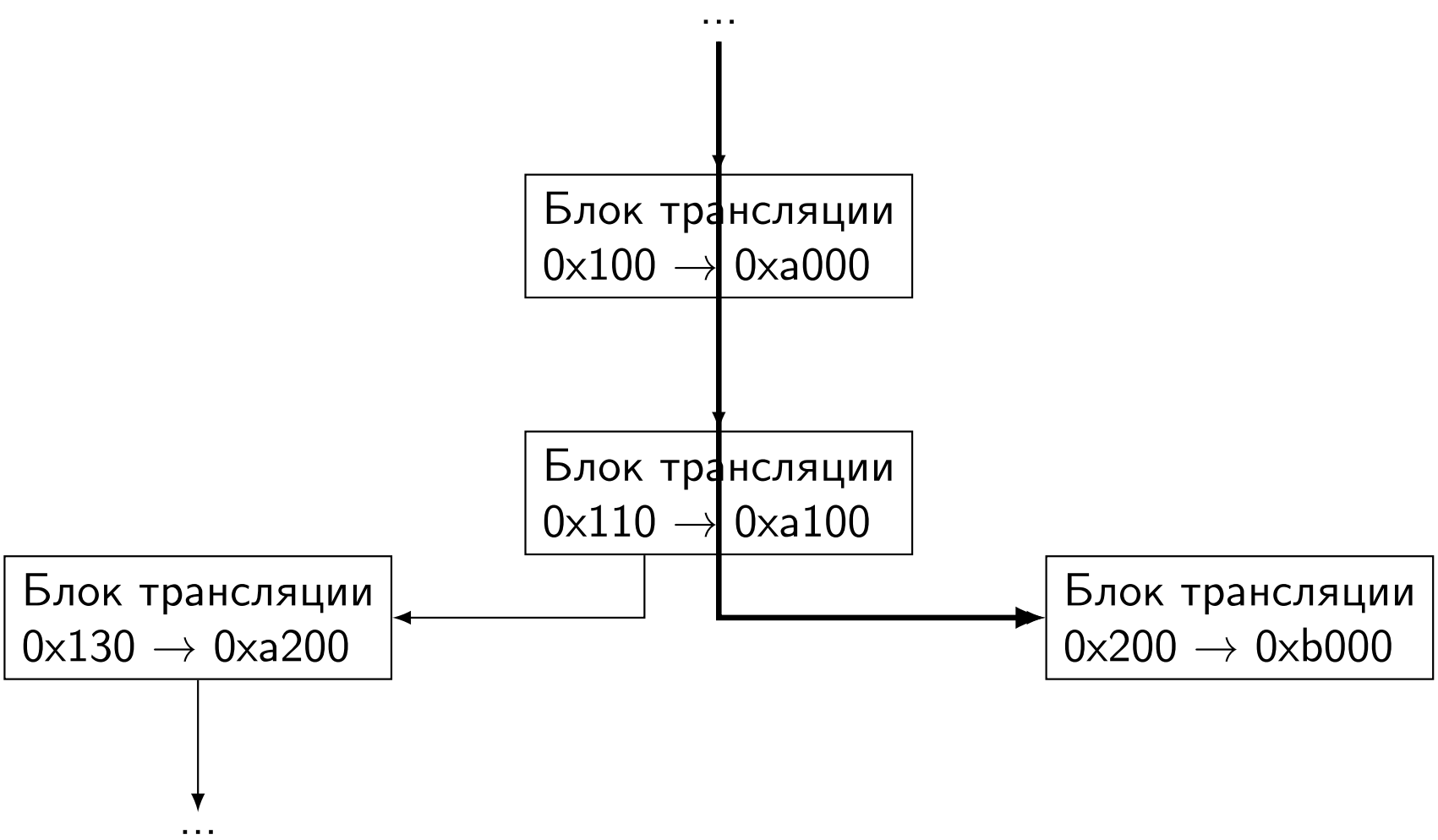
Опережающая трансляция



Опережающая трансляция

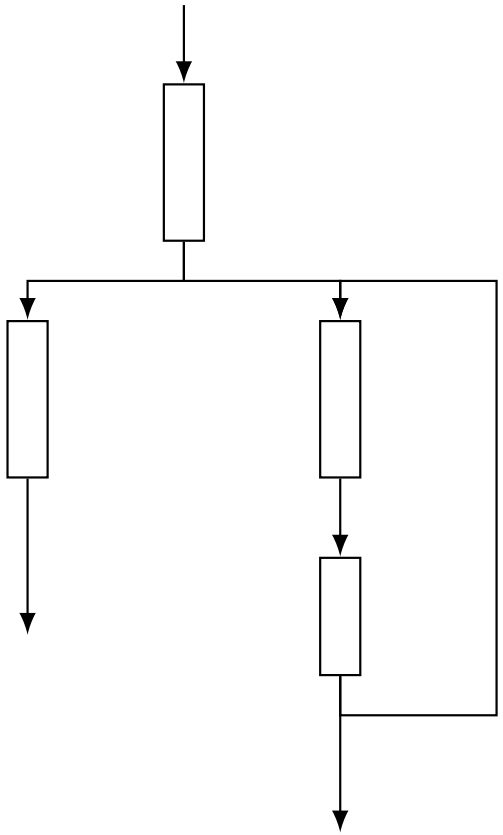


Опережающая трансляция



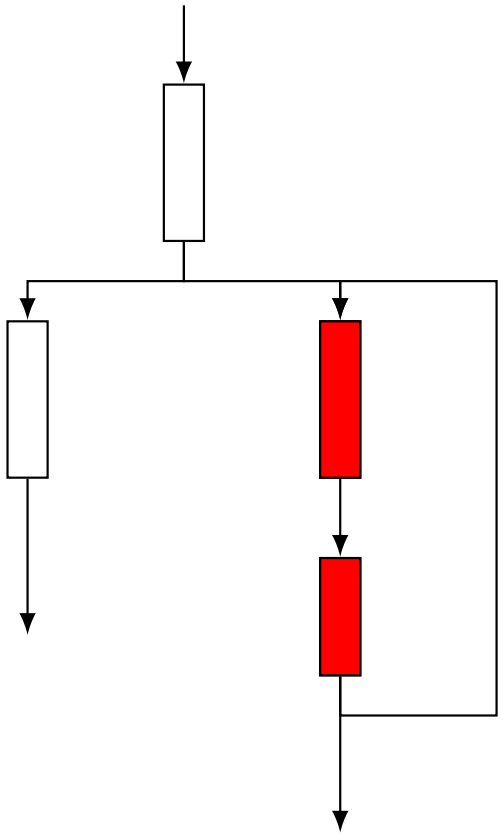
Трансляция горячего кода

Исполнение 1: интерпретация с трассированием



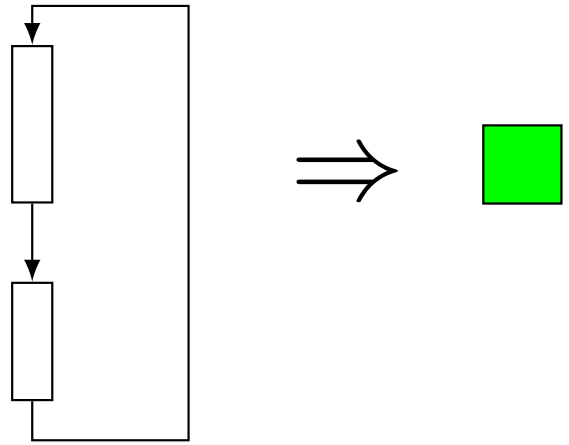
Трансляция горячего кода

Исполнение *N*: обнаружение «горячих» участков трассы



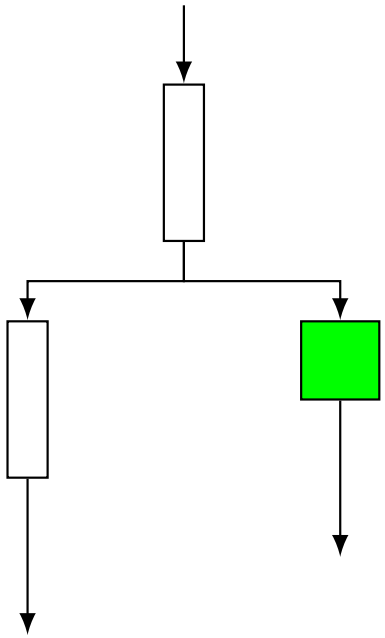
Трансляция горячего кода

Трансляция трассы в блоки



Трансляция горячего кода

Подстановка новых блоков в трассу

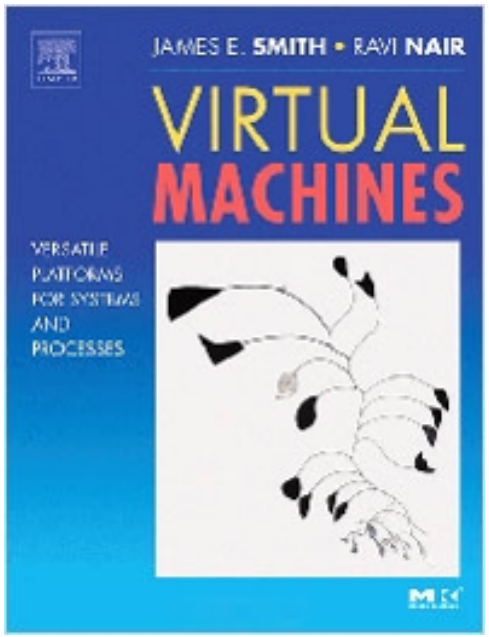


Итоги

- Интерпретация, компиляция, трансляция
- Двоичная трансляция
- Статическая и динамическая трансляция
- Шаблон, капсула
- Промежуточное представление
- Самомодифицирующийся код
- Обнаружение кода
- Сложности оптимизации кода при ДТ

Литература I

Jim Smith and Ravi Nair. Virtual Machines: Versatile Platforms for Systems and Processes. Morgan Kaufmann - 2005



Литература II

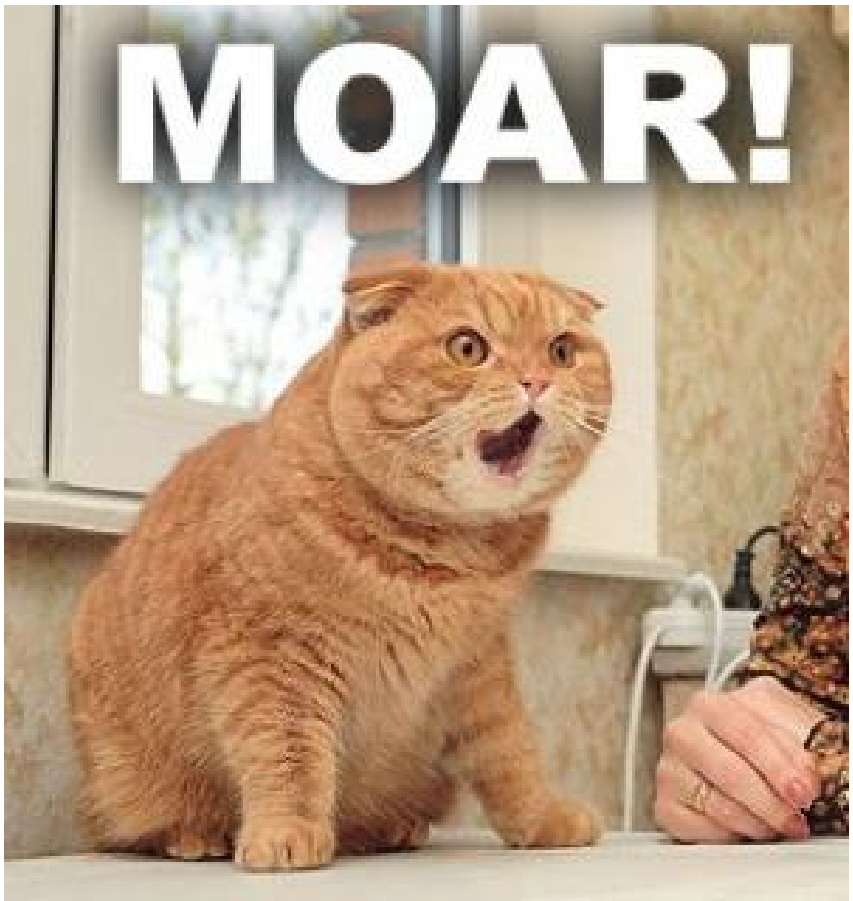
- Fabrice Bellard. QEMU, a Fast and Portable Dynamic Translator http://www.usenix.org/publications/library/proceedings/usenix05/tech/freenix/full_papers/bellard/bellard.pdf
- Anton Chernoff and Ray Hookway. DIGITAL FX!32 Running 32-Bit x86 Applications on Alpha NT http://www.usenix.org/publications/library/proceedings/usenix-nt97/full_papers/chernoff/chernoff.pdf

Литература III

Leonid Baraz [et al.] IA-32 Execution Layer: a Two-Phase Dynamic Translator Designed to Support IA-32 Applications on Itanium®-Based Systems.
http://www.microarch.org/micro36/html/pdf/goldenberg-IA32ExecutionLayer.pdf

На следующей лекции

Ещё быстрее! Прямое исполнение



Спасибо за внимание!

Слайды и материалы курса доступны по адресу
http://is.gd/ivuboc

Замечание: все торговые марки и логотипы, использованные в данном материале, являются собственностью их владельцев. Представленная здесь точка зрения отражает личное мнение автора, не выступающего от лица какой-либо организации.