ФРТК

iSCALARE

intel

Лаборатория суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур

Двоичная трансляция и симуляция

Григорий Речистов

grigory.rechistov@phystech.edu

17.03.2014

- Статическая ДТ
- Динамическая ДТ
- Проблемы и решения

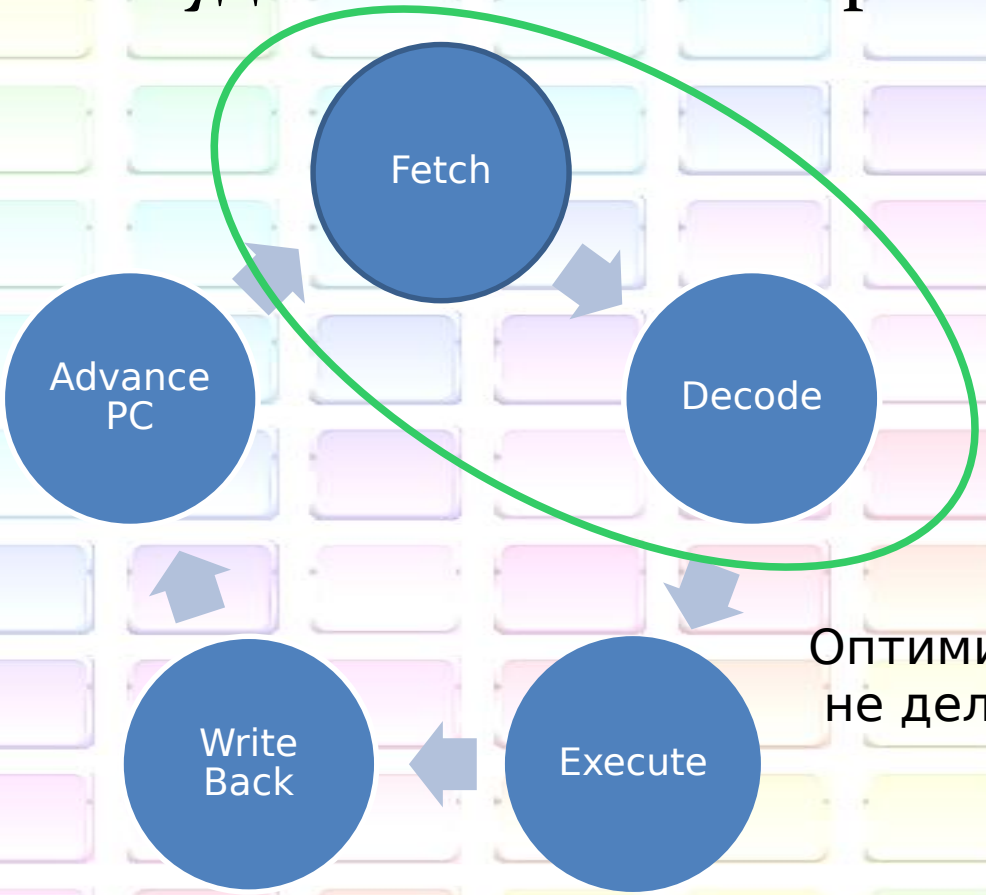
Вопросы к прошлой лекции

1. Определение процесса декодирования
2. Определение байта
3. Что такое преобразование адресов?
4. Что такое MMIO и зачем оно используется?

На предыдущих лекциях:

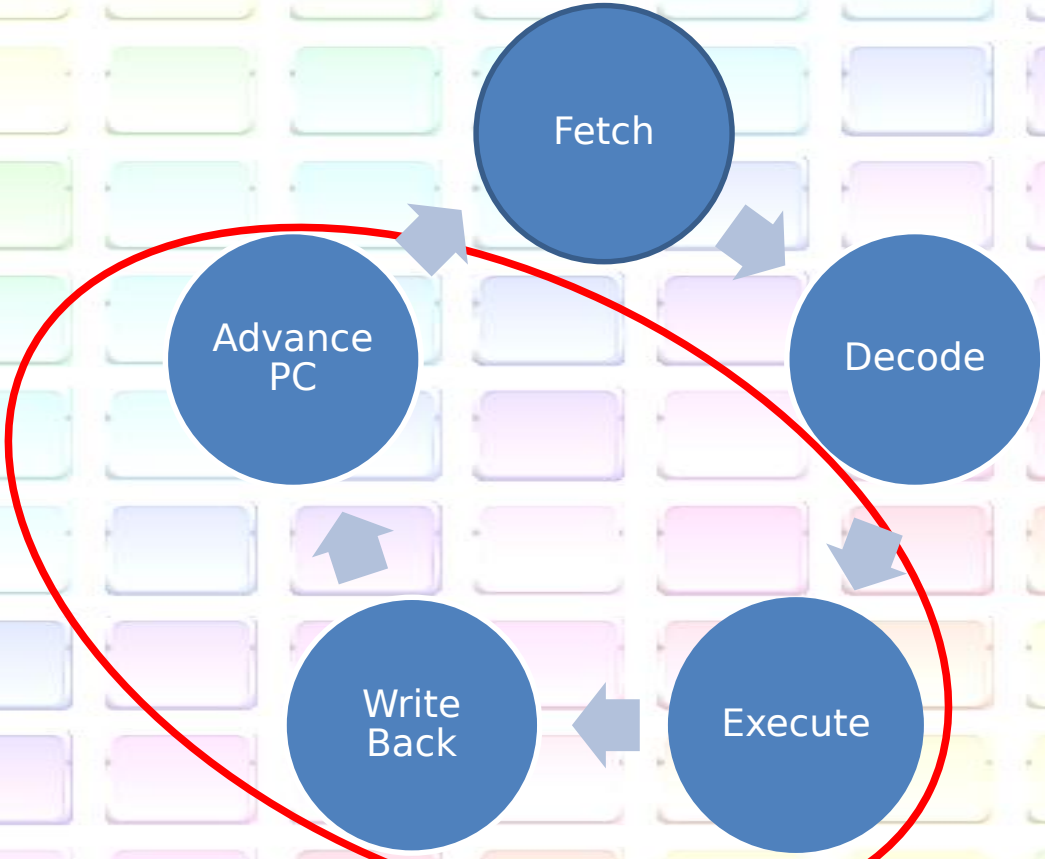
- Интерпретация выражается через симуляцию основного цикла процессора
- В своей простейшей форме она обеспечивает малую скорость модели
- Улучшения интерпретатора основаны на повторном использовании предыдущих результатов (декодирования)

Что нам удалось оптимизировать



Оптимизировать — не делать вообще

Что бы нам ещё улучшить?



Языки высокого уровня, компиляция

- Basic — большинство существующих реализаций являются интерпретаторами
  - Прочитал строку — распознал команды — исполнил
  - Хорошо, но медленно
- FORTRAN, C, Pascal. Работа в два прохода
  1. Компилятор преобразует исходный код в машинный (опционально применяет оптимизации)
  2. Машинный код исполняется

Трансляция гостевого машинного кода в хозяйский

Входной язык —
гостевой машинный код

1. Трансляция

Целевой язык —
хозяйский машинный код

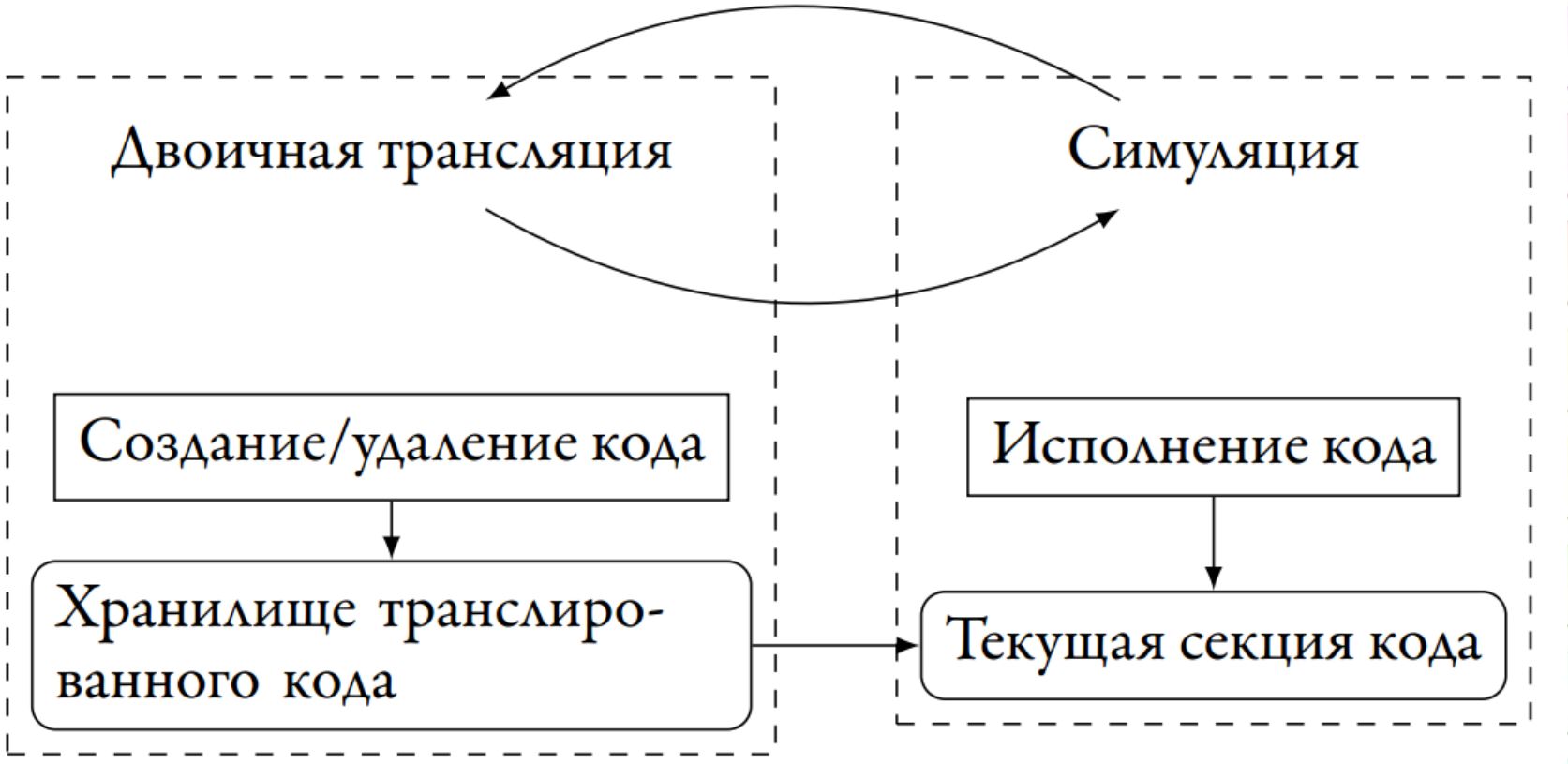
```
Disassembly of section .init:

0000000000401fd0 <_init>:
  401fd0:       48 83 ec 08             sub    $0x8,%
  401fd4:       e8 f7 25 00 00          callq  4045d0
  401fd9:       e8 82 26 00 00          callq  404660
  401fde:       e8 dd f7 00 00          callq  4117c0
  401fe3:       48 83 c4 08             add    $0x8,%
  401fe7:       c3                      retq

Disassembly of section .plt:

0000000000401ff0 <__ctype_toupper_loc@plt-0x10>:
  401ff0:       ff 35 fa 6f 21 00       pushq  0x21
  401ff6:       ff 25 fc 6f 21 00       jmpq   *0x21
  401ffc:       0f 1f 40 00             nopl   0x0(%ra
```

2. Исполнение

Двоичная трансляция (Binary translation)
ДТ (BT)

- Перевод кода из гостевого ISA в *эквивалентный* код хозяйского ISA
- Результаты трансляции можно переиспользовать (если они сохранены и валидны)
- Исполнение не испытывает замедления на «родной» аппаратуре

Фазы ДТ


Двоичная трансляция
Симуляция
Создание/удаление кода
Исполнение кода
Хранилище транслиро-
ванного кода
Текущая секция кода

Алгоритм

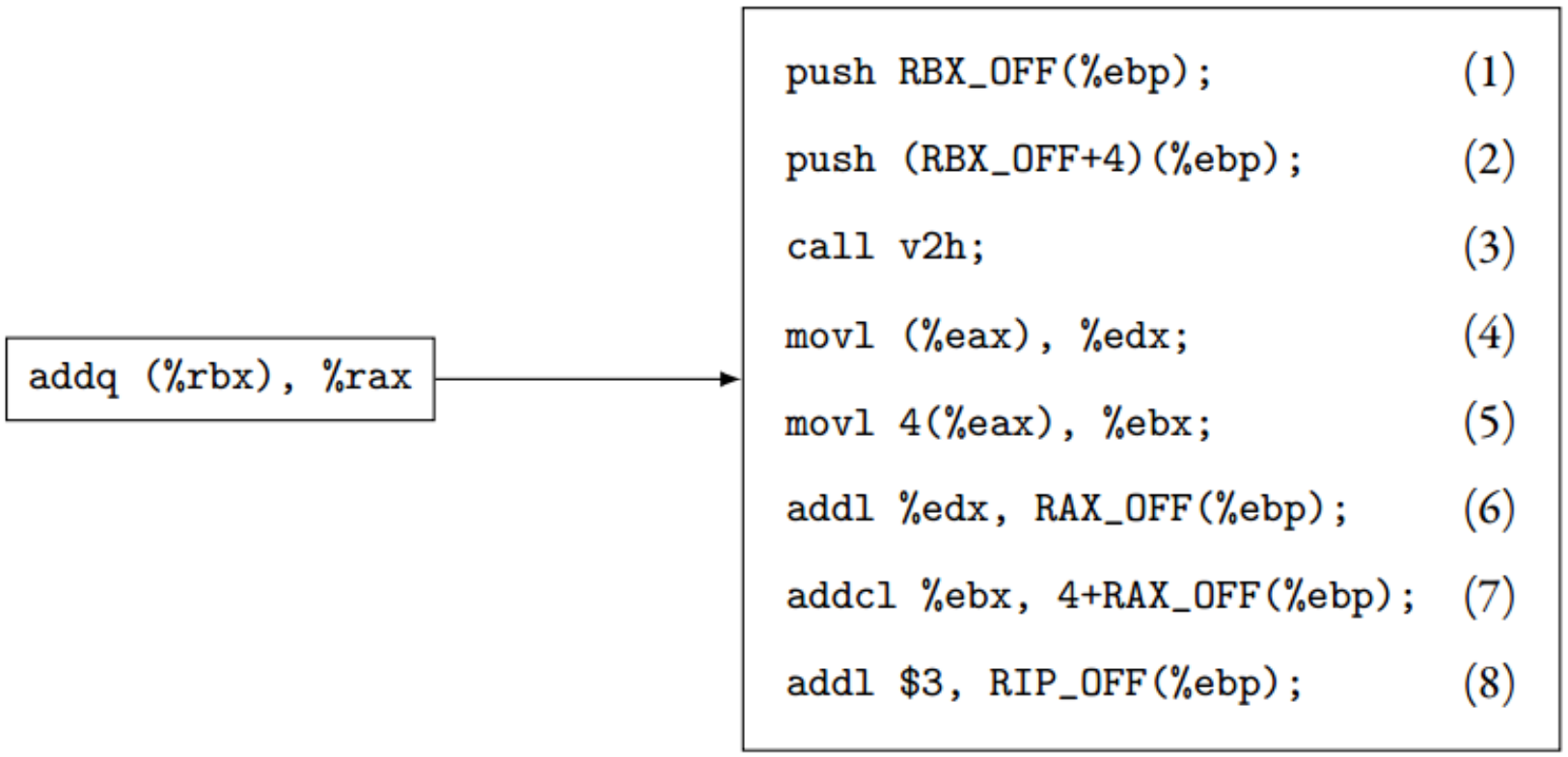
```
translate() {
PC = start_addr;
bufptr = start_buf;
while (! enough) {
   instr = fetch(PC);
   opcode = decode(instr);
   capsule = capsules[opcode];
   memcpy(capsule, bufptr);
   PC ++;
   bufptr ++;
};
memcpy(return_jmp, bufptr);
};
```

```
execute(){
   setjmp(back);
   goto start_buf;
   back: ;
};
```

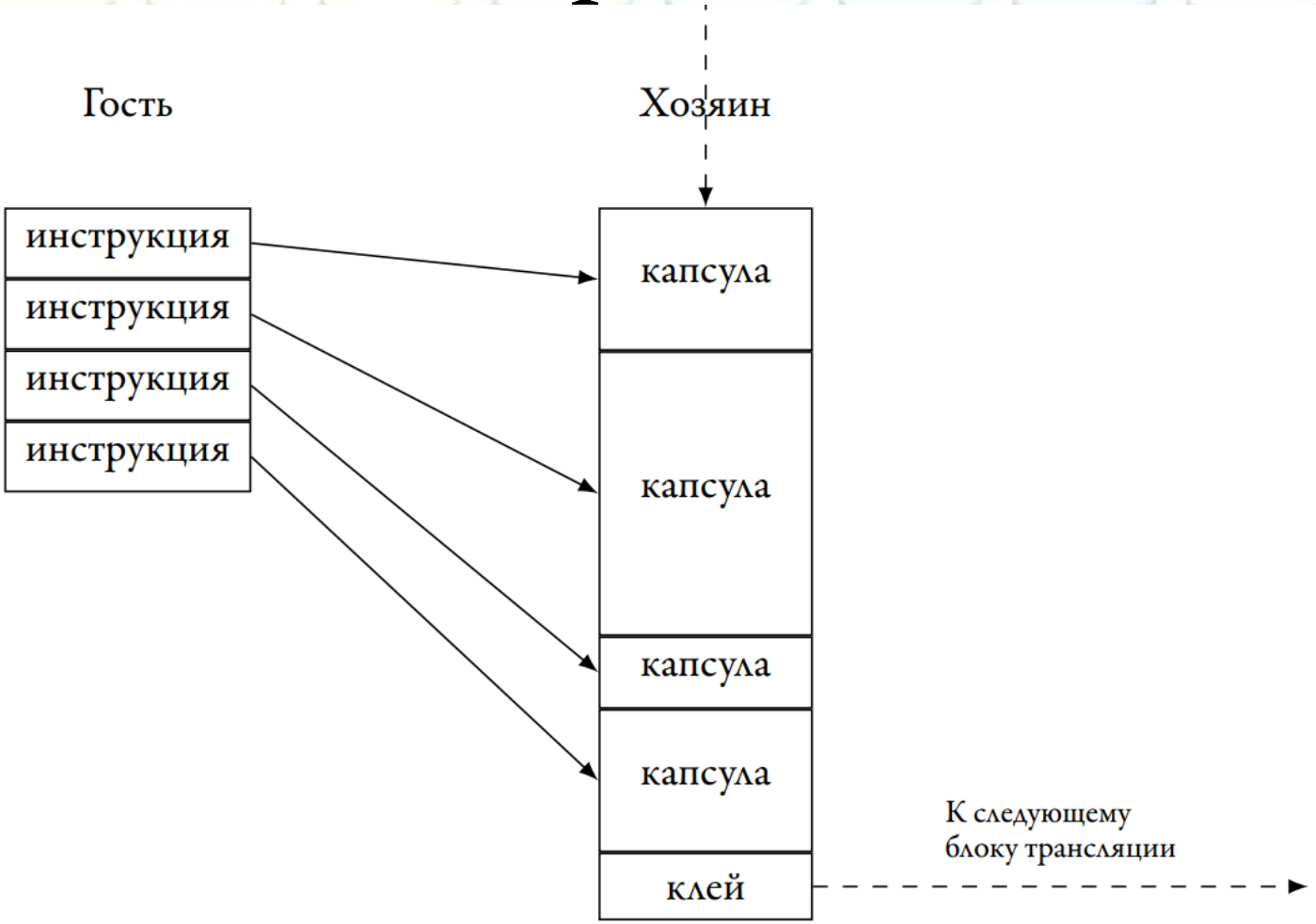
Капсула

Гостевой код, архитектура IA-32 EMT (64 бит)

Хозяйский код, архитектура IA-32 (32 бит)



Блоки трансляции



Статическая ДТ

- Стадия ДТ исполняется заранее, до исполнения
- Результат ДТ сохраняется на диске
- Мы можем применить агрессивные многопроходные оптимизации
- Пример: *Digital FX!32*. **IA-32 → Alpha**
  http://www.usenix.org/publications/library/proceedings/usenix-nt97/full_papers/chernoff/chernoff.pdf
- Бонус-сценарий: двоичная оптимизация

Динамическая ДТ

- Происходит непосредственно во время симуляции, результат хранится в памяти
- Фаза ДТ чередуется с исполнением → не может быть длительной
- ДТ ограничена в оптимизациях
- Может обработать самомодифицирующийся код
- Корректная полноплатформенная статическая ДТ *невозможна* без механизма поддержки времени исполнения, т. е. динамической ДТ

Оптимизация результатов трансляции


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

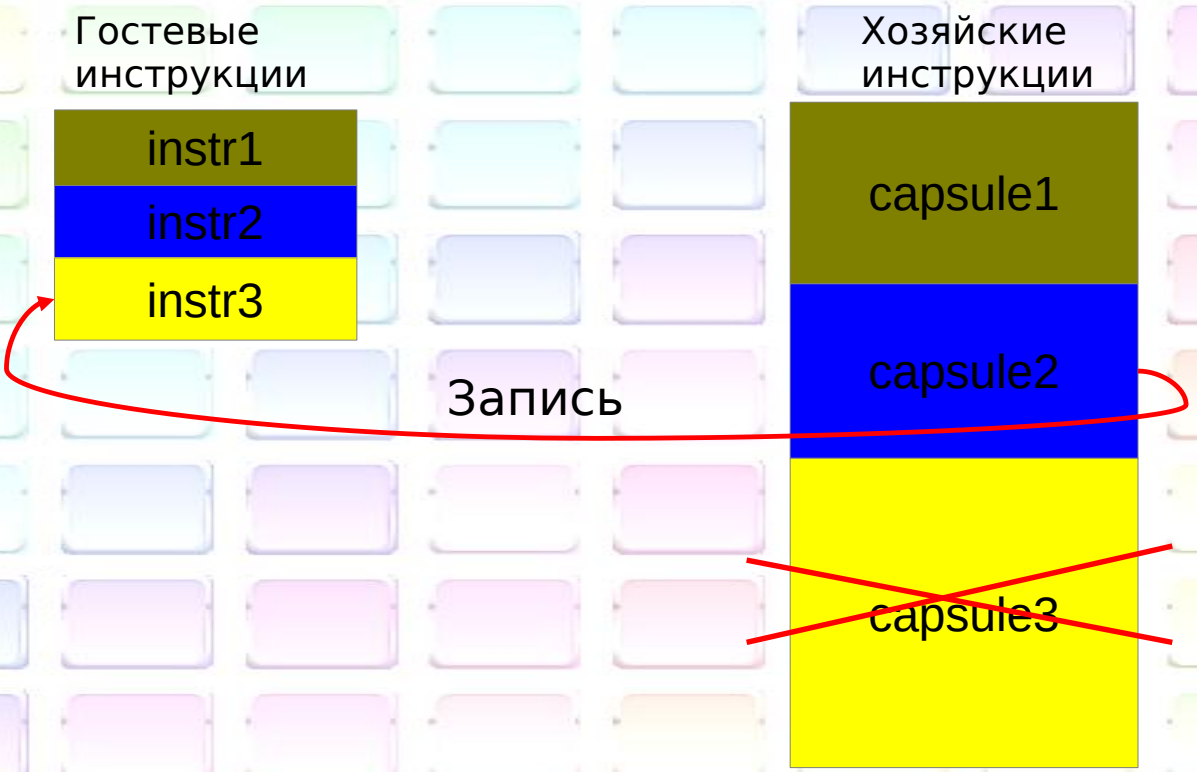
```
<instr1>
inc PC_OFF(%r14)
<instr2>
inc PC_OFF(%r14)
<instr3>
inc PC_OFF(%r14)
<instr4>
inc PC_OFF(%r14)
<instr5>
inc PC_OFF(%r14)
<branch>
```

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

Почему оптимизации при ДТ затруднительны

- В отличие от ЯВО, машинный код содержит меньше информации об исходном алгоритме
- Мы не можем делать многие предположения, необходимые для компиляторных оптимизаций без нарушения корректности
  - Адреса переменных — их нет
  - Границы процедур — их нет
  - Адреса переходов — известна только часть из них

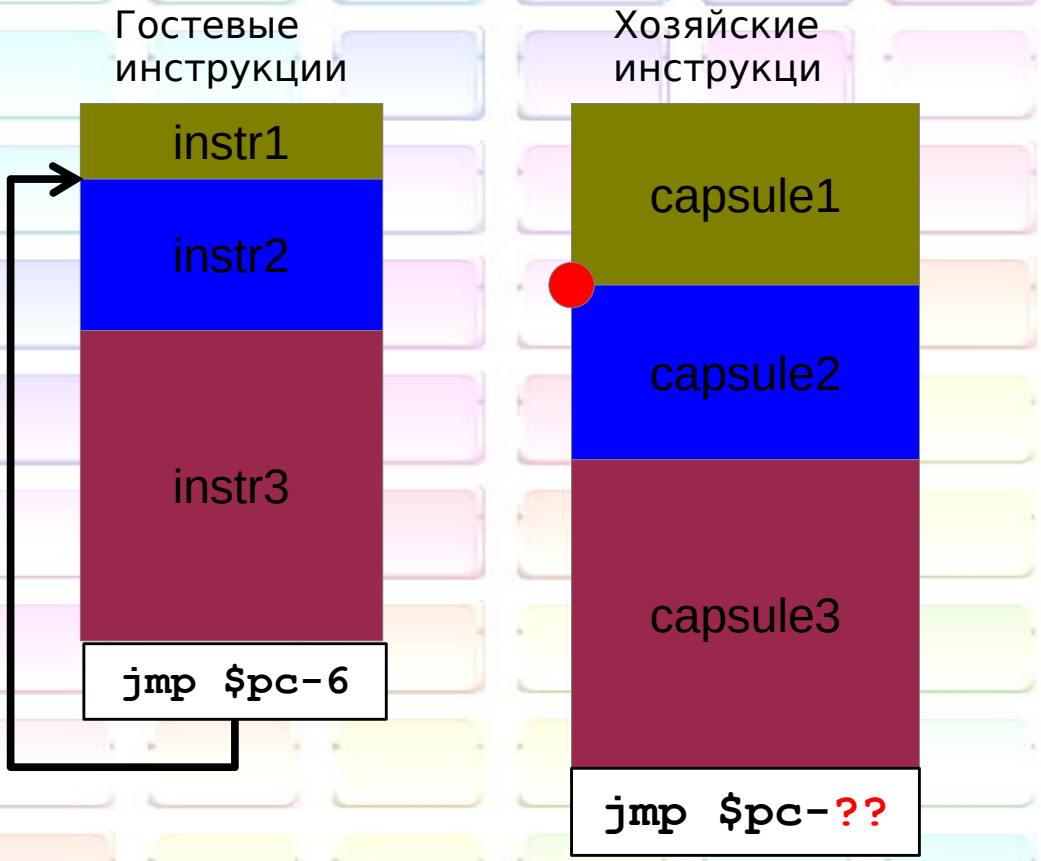
Самомодифицирующийся код (self modifying code, SMC)



Обнаружение кода (1/2)

Code discovery problem

- Найти границы инструкций
- Отличить код от данных



Обнаружение кода (2/2)

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

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

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

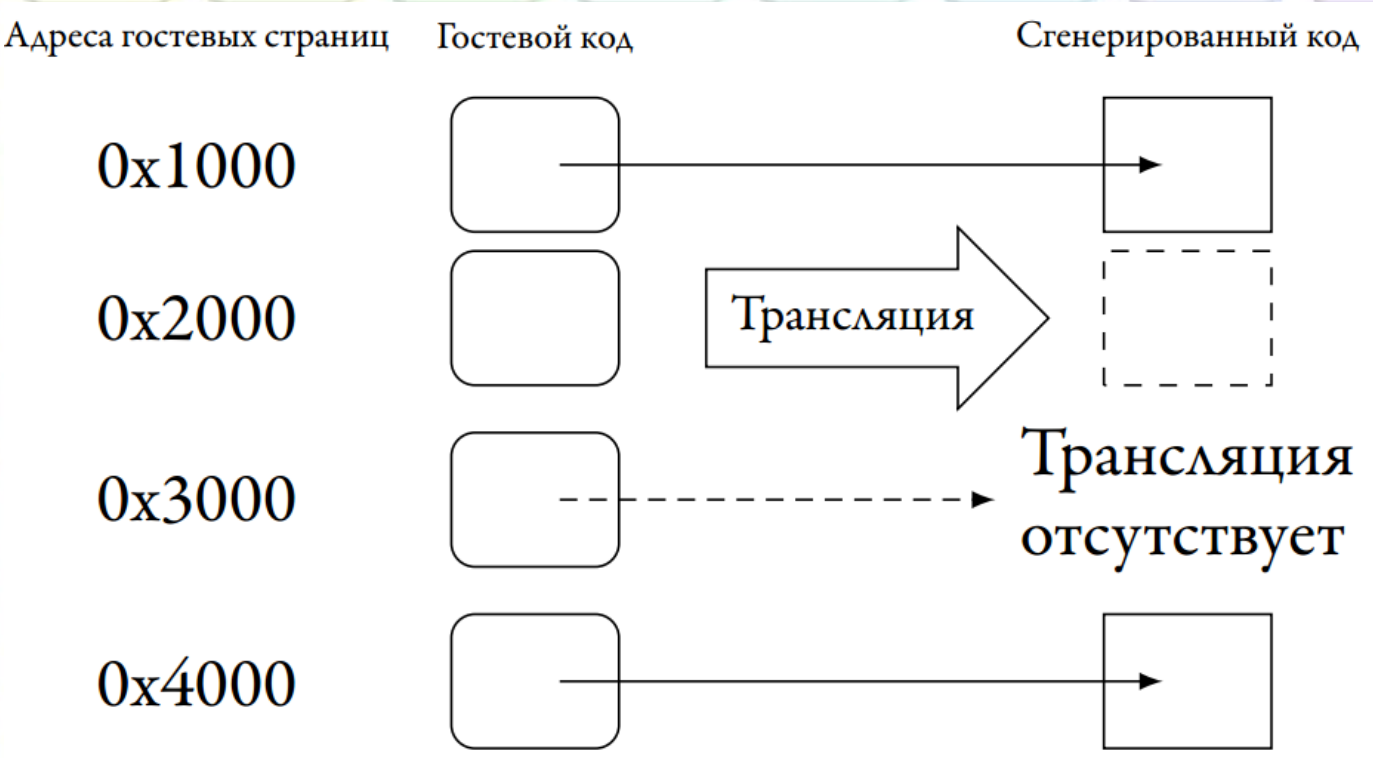
Блоки для трансляции

```
while (! enough) {…}
```

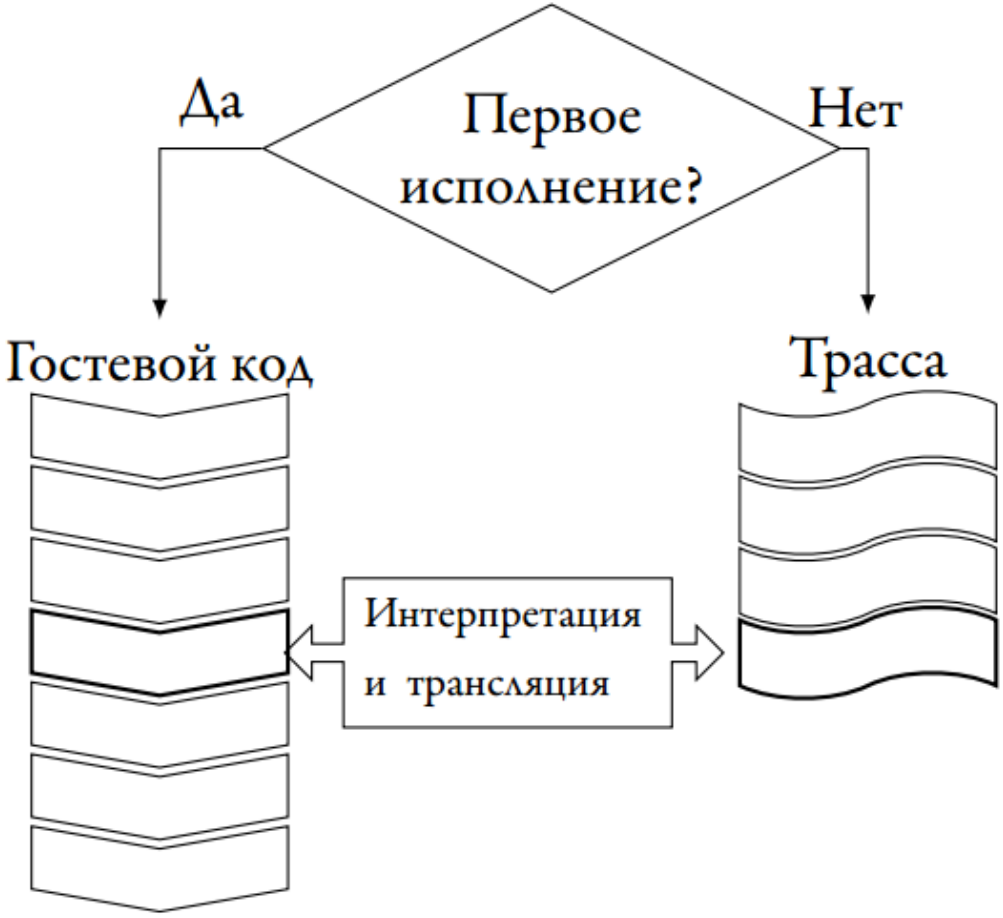
Чем ограничивать длину блоков трансляции?

1. Гостевая страница
2. Цепочка инструкций (трасса), исполнявшаяся ранее

Гостевая страница



Трасса ранее исполнявшихся инструкций



И ещё:

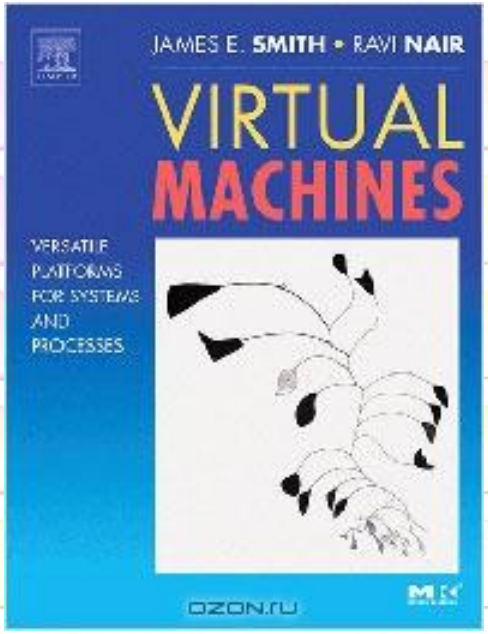
- Трансляция становится неактуальной при изменении режима процессора
- Смысл машинного кода изменяется
- Но: можно хранить результаты ДТ с ассоциированным режимом, для которого они валидны
- Иметь несколько трасс для одного и того же региона памяти
- Можно шарить кэш трансляций между несколькими гостевыми ЦПУ

Итоги

- Интерпретация, компиляция (трансляция)
- Двоичная трансляция. Статическая, динамическая трансляция
- Капсула
- SMC
- Code discovery
- (Не)возможность оптимизации кода при ДТ

Рекомендуемая литература (1/2)

Jim Smith, Ravi Nair. Virtual Machines: Versatile Platforms for Systems and Processes. 2005



Рекомендуемая литература (2/2)

Fabrice Bellard. **QEMU, a Fast and Portable Dynamic Translator**

- http://www.usenix.org/publications/library/proceedings/usenix05/tech/freenix/full_papers/bellard/bellard.pdf

Mathieu Brethes. **Atome - Binary Translation for Accurate Simulation**

- http://www8.cs.umu.se/education/examina/Rapporter/MathieuBrethes.pdf

Nigel Topham, Daniel Jones. **High Speed CPU Simulation using JIT Binary Translation**

- http://homepages.inf.ed.ac.uk/npt/pubs/mobs-07.pdf

На следующей лекции:

- Ещё быстрее!
- Прямое исполнение
- Аппаратная поддержка

Спасибо за внимание!

Все материалы курса выкладываются на сайте лаборатории:
http://iscalare.mipt.ru/material/course_materials/