Платформа для машинно-независимого исполнения программного кода с возможностью JIT-компиляции

Автор: Пимкин Артем

Научный руководитель: И. Р. Дединский

2013

Цель и задачи проекта

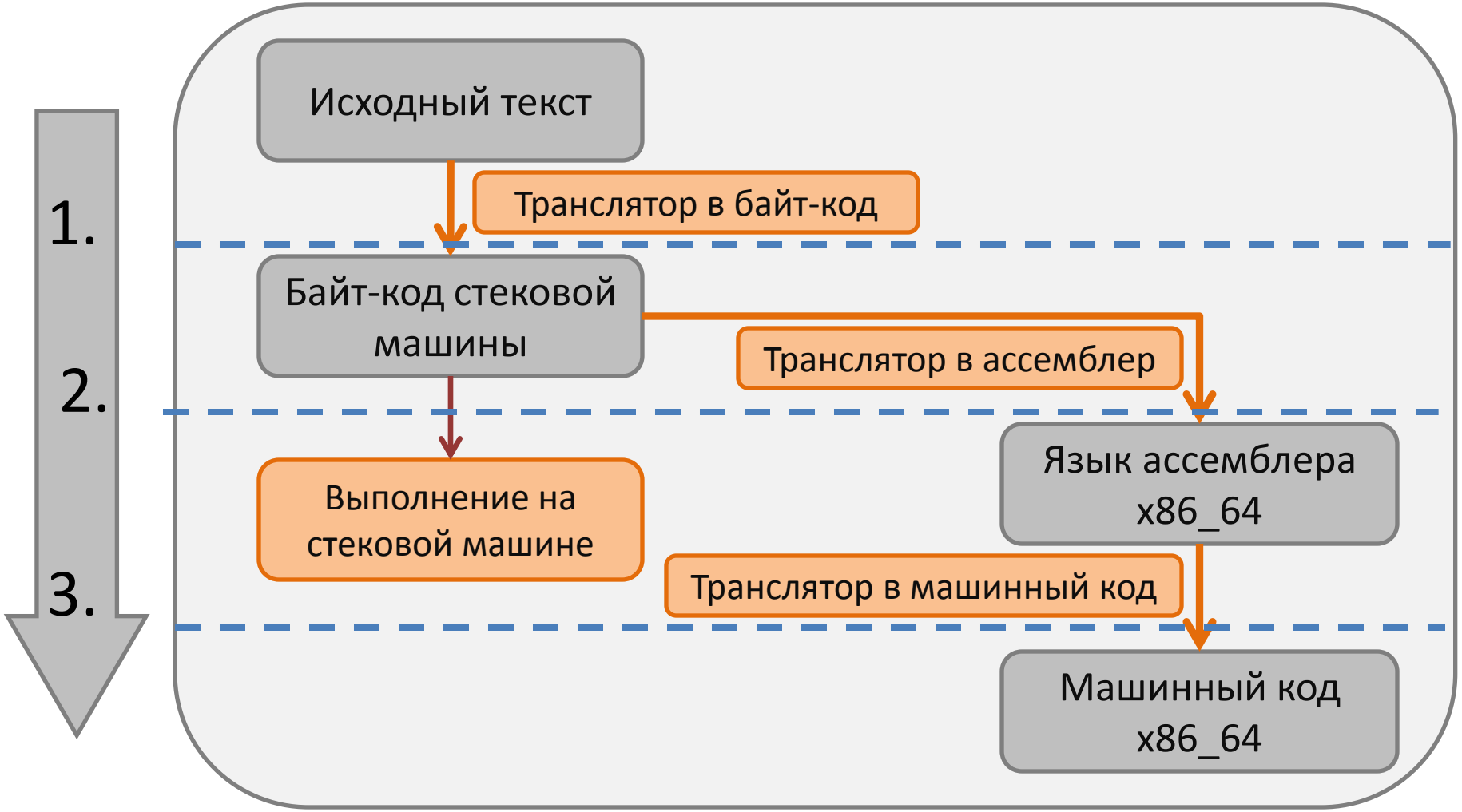
Цель работы – разработка платформы для машинно-независимого исполнения программного кода с возможностью JIT-компиляции для архитектуры IA-32E.

Задачи проекта:

- Разработать программу, выполненную в соответствии с **модульным подходом,** работающую с программным кодом, который задан изменяемым командным набором.
- Создать модуль, **непосредственно выполняющий код** в формате внутреннего представления языка, фактически, виртуальную стековую машину.
- Создать модуль, реализующий **трансляцию байт-кода стековой машины в** в язык ассемблера x86_64.
- Разработка **модуля JIT-компиляции** языка ассемблера x86_64.
- Разработка удобного интерфейса обращения к JIT-компилирующему модулю, позволяющего менять JIT-модуль **без изменений** в коде остальных модулей.

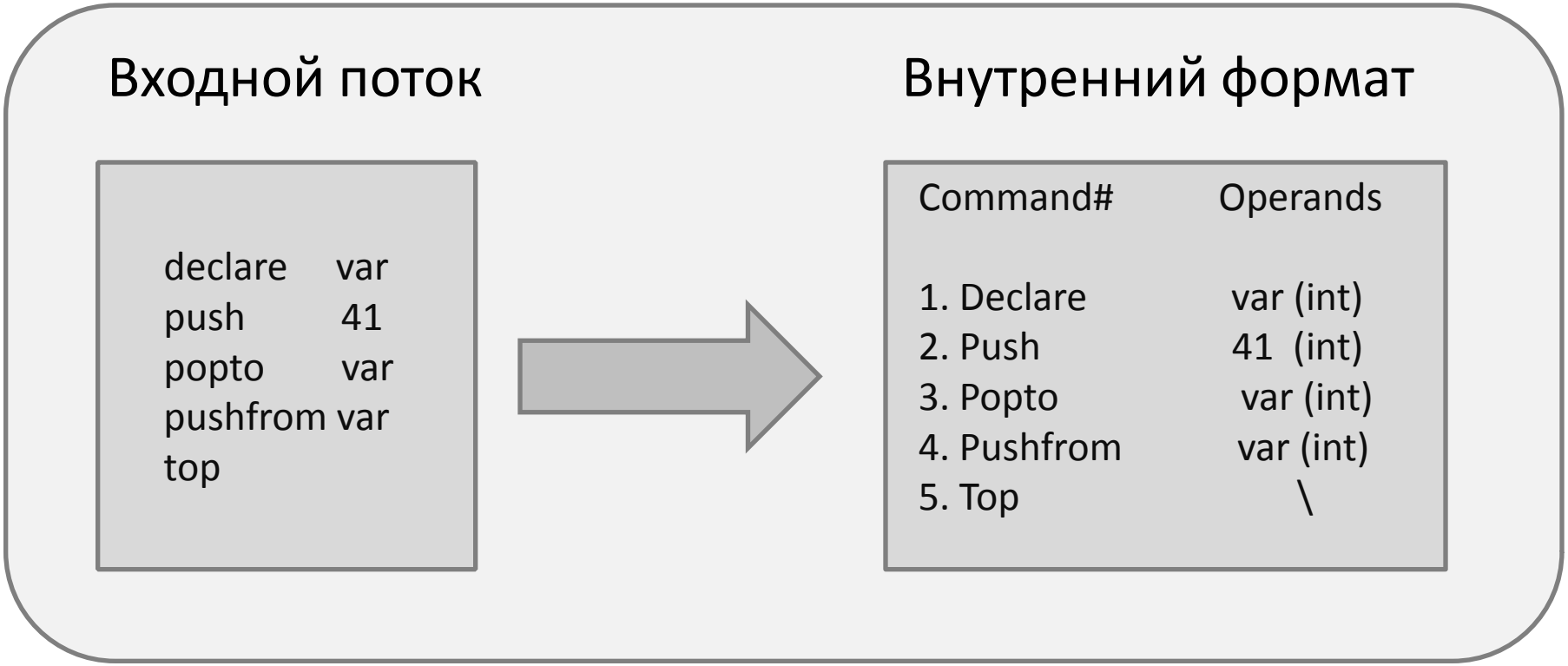
Модульно-конвейерный подход

Исходный текст проходит конечное число этапов преобразования. За каждый этап отвечает отдельный модуль.



Трансляция во внутреннее представление

- Считывание программного кода
- Разбиение программного кода команды
- Определение операндов и их типов для каждой команды
- Запись во внутреннее представление
- Формат внутреннего представления совпадает с байт-кодом стековой машины



Виртуальная стековая машина

- Каждой команде из изменяемого командного набора соответствует некоторая функция
- Стековая машина обрабатывает входной список команд, вызывая для каждой команды соответствующую ей функцию

Список команд

Command#	Operands
1. Declare	var (int)
2. Push	41 (int)
3. Popto	var (int)
4. Pushfrom	var (int)
5. Top	\

Исполняющие функции

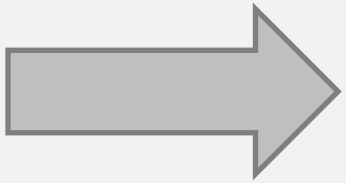
```
…
void Executor::Push ()
{
   stack -> push (execCmds[executingCmd]
                        -> intArgs [0]);
}
…
void Executor::Top ()
{
   int var = stack -> pop ();
   prinf (“%d”, var);
   stack -> push (var);
}
```

JIT-компилятор

- JIT-компилятор - это модуль, способный заменить виртуальную стековую машину
- Каждой команде из изменяемого командного набора соответствует некоторое множество инструкций языка ассемблера, аналогичное по функционалу

Байт-код стековой машины

Command#	Operands
1. Push	41 (int)
2. Push	100 (int)
3. Add	\
4. Top	\



Код x86_64

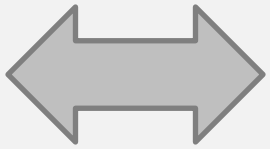
```
push  41
push 100
pop   rax
pop   rbx
add   rax, rbx
push rax
mov  rdi, rsp
mov  rdx, 0x42f5eb
call    rdx
```

JIT-компилятор

- JIT-компилирующая библиотека языка ассемблера - модуль. Изначально использовалась сторонняя библиотека, которая позже была заменена на собственную
- Создается область памяти с необходимыми правами доступа
- Производится трансляция кода на языке ассемблера в последовательность байт, выполняемую непосредственно процессором

Код x86_64

```
push  41
push 100
pop   rax
pop   rbx
add   rax, rbx
push rax
mov  rdi, rsp
mov  rdx, 0x42f5eb
call    rdx
```



Машинный код x86_64

```
48 68 29 00 00 00
48 68 64 00 00 00
48 8f c0
48 8f c3
48 03 c3
48 ff f0
48 8b fc
48 ba eb f5 42 00 00 00
00 00
48 ff d2
```

Результаты

- **Разработана программа**, выполненная в соответствии с модульным подходом, работающая с программным кодом, который задан изменяемым командным набором.
- **Создан модуль**, непосредственно **выполняющий код** в формате внутреннего представления языка, фактически, **виртуальная стековая машина**.
- **Создан модуль**, реализующий **трансляцию** кода в формате внутреннего представления языка в **язык ассемблера x86_64**.
- Разработан модуль **JIT-компиляции** языка ассемблера x86_64.
- Разработан удобный **интерфейс обращения** к JIT-компилирующему модулю, позволяющий менять JIT-модуль без изменений в коде остальных модулей.

Благодарности

- Работа была выполнена в рамках учебного задания экспериментальной площадки по проектной деятельности И. Р. Дединского

Использованная литература

- Intel 64 and IA 32 Software Developers manual
- Материалы с сайта stackoverflow.com
- С. В. Зубков. Assembler для DOS, Windows и UNIX.

СПАСИБО ЗА ВНИМАНИЕ!