Разработка модульной системы трансляции с возможностью обратимой трансляции для ЯВУ, машинно-независимым исполнением программ и поддержкой JIT-компиляции

Данила Байгушев, 8-9 класс

Научный руководитель: И. Р. Дединский

УРОВНИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Smalltalk

```
transcript show:
'Hello
      world'
```

C

```
printf("Hello
      world");
```

Asm

```
mov ah, 9
lea dx, Hw
int 21h
...
Hw db 'Hello
     world$'
```

СВЕРХ-ВЫСОКИЙ УРОВЕНЬ

ВЫСОКИЙ УРОВЕНЬ

НИЗКИЙ УРОВЕНЬ

?

...

LOLcode

```
VISIBLE "Hello
        World!"
```

Piet

Brainfuck

```
++++++++++[>++++
+++>++++++++++>+
++>+<<<<]>++.
```

Shakespeare

```
"Simple Shakespeare program"
      Dramatic Personae
Romeo,  young man.
Juliet, woman of grace.
            Act I.
           Scene I.
  [Enter Romeo and Juliet]
Juliet:
    You are big hero.
    Speak your mind!
Romeo:
    You are the sum of your
    beautiful kingdom and
    warm peaceful sunny
    summer's day.
    Speak your mind!
           Scene II.
Romeo:
    Speak your mind.
    You are as worried as
    small smooth hamster.
Juliet:
    Speak YOUR mind.
[Exeunt]
```

Машинный код

```
BB 11 01 B9 0D 00 0E 8A 07 43 CD 10 E2 F9 CD 20 48 65 6C 6F 2C 20 57 6F 72 6C 64 21
```

ЦЕЛЬ И ЗАДАЧИ

Цель

Разработка модульной расширяемой системы трансляции с возможностью обратимой трансляции для случая ЯВУ$_1$ ↔ ЯВУ$_2$, машинно-независимого исполнения программ и поддержкой JIT-компиляции для Intel x86.

Задачи

Разработать

- Модульную архитектуру для возможности расширения системы
- Высокоуровневый C-подобный скриптовый язык
- Расширенную модификацию языка программирования «Шекспир» с поддержкой вызова функций
- Формат AST для единого внутреннего представления программного кода

Реализовать

- Транслятор с разработанной модульной архитектурой
- Софт-процессор для исполнения оттранслированных программ
- Вариант софт-процессора с возможностью JIT-компиляции в машинный код семейства процессоров Intel x86 (64-bit)

ТРАНСЛЯЦИЯ ЯЗЫКОВ НИЗКОГО УРОВНЯ

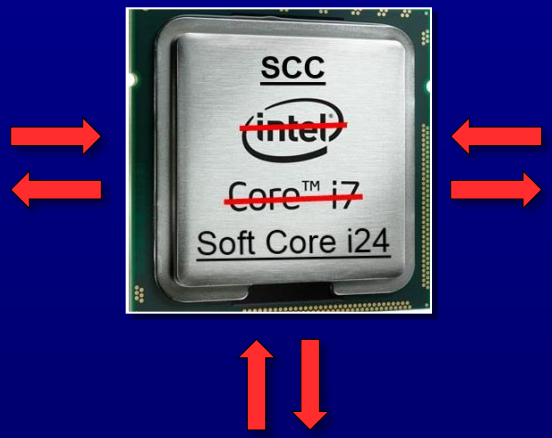
Текст на Asm

```
push 4
pushr a ; reg 0
pushr c ; reg 2

mul
mul

pusha b ; addr 0
pusha b ; addr 0

mul

sub

sqrt
```

Компиляция

Машинный код

0x00	4	0x01	0x10	0x01	0x11	0x20	0x20	0x02	0x20	0x02	0x20	0x20	0x21	0x22



Память программы

2	0x00
32	0x01
-4675	0x02
23	0x03
43783	0x04
-5674	0x05
33464	0x06
4366	0x07

Стек

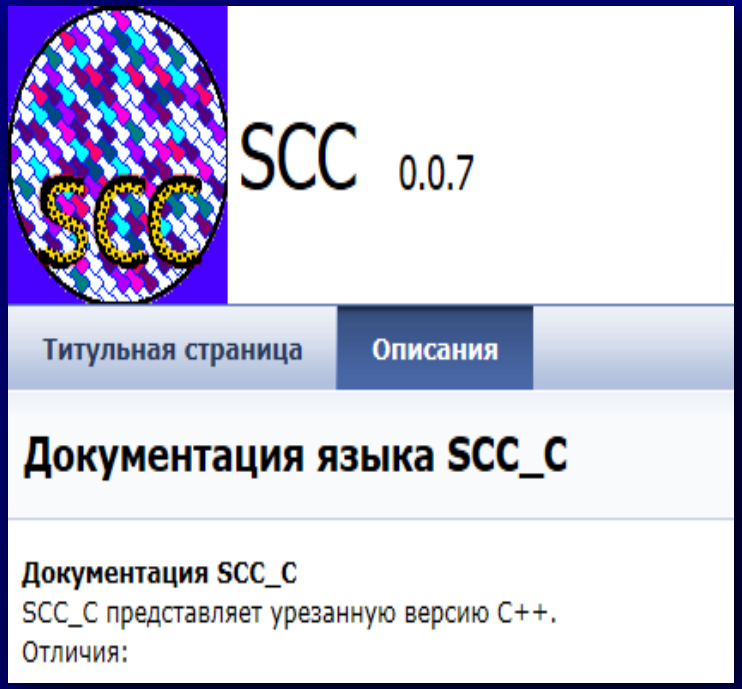
0	0x00
4	0x01
2	0x02
-3423	0x03
23783	0x04
2323	0x05
-464	0x06
236	0x07

Регистры

1	231	1	232
0x00	0x01	0x02	0x03
a	b	c	d

ЯЗЫК SCC_C

SCC_C - подмножество языка C

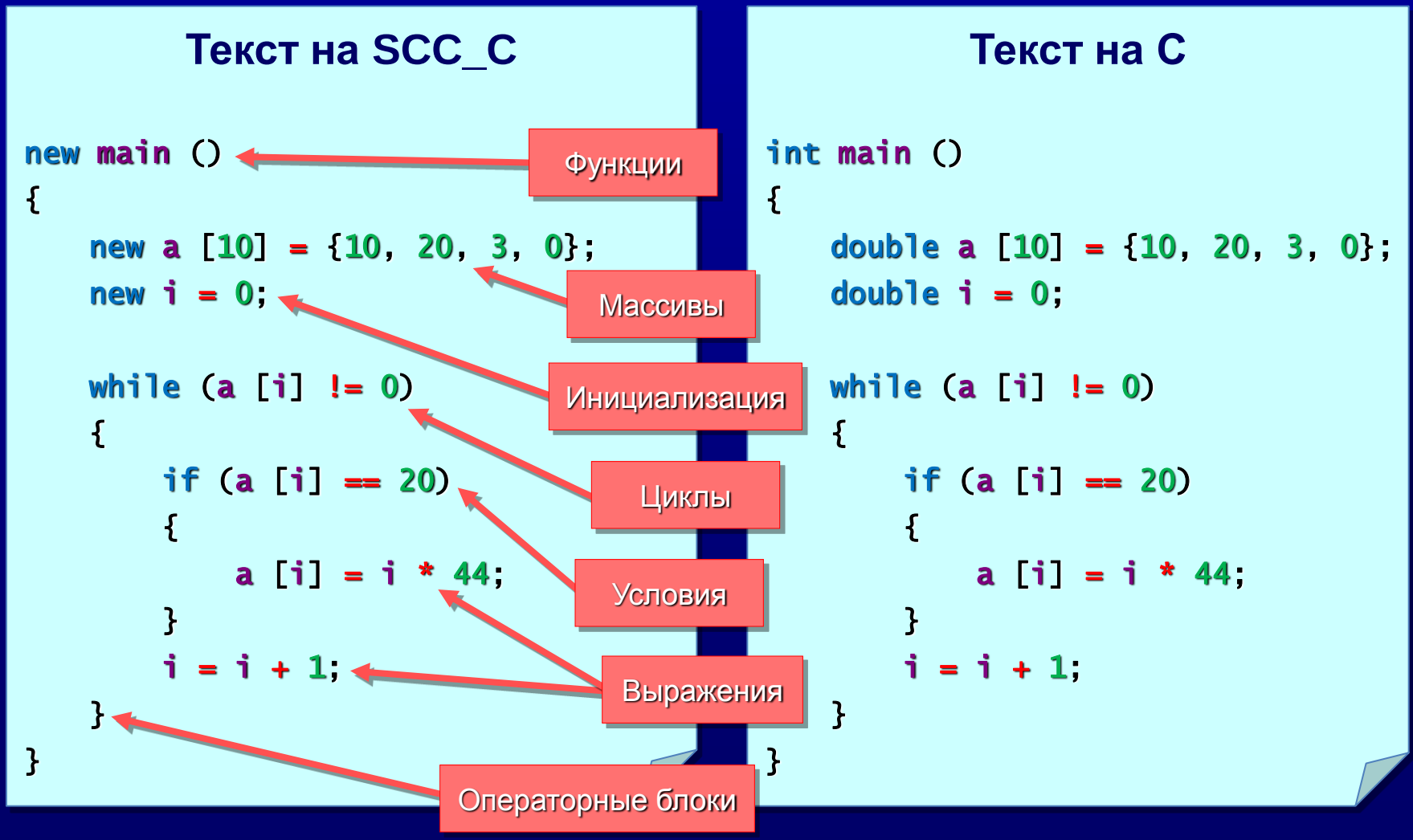


Полный список отличий см. в документации по SCC

Текст на SCC_C

```
new a = 0;  // Создание переменной

if (a == 0)
{
    a = (22 + 33) * 44;
}
else
{
    a = 22 + 33 * 44;
}

out a;  // Вывод на экран
```

ЯЗЫК SCC_C


Текст на SCC_C
new main ()
{
new a [10] = {10, 20, 3, 0};
new i = 0;
while (a [i] != 0)
{
if (a [i] == 20)
{
a [i] = i * 44;
}
i = i + 1;
}
}
Текст на C
int main ()
{
double a [10] = {10, 20, 3, 0};
double i = 0;
while (a [i] != 0)
{
if (a [i] == 20)
{
a [i] = i * 44;
}
i = i + 1;
}
}
Функции
Массивы
Инициализация
Циклы
Условия
Выражения
Операторные блоки

ТРАНСЛЯЦИЯ ЯЗЫКОВ ВЫСОКОГО УРОВНЯ

Текст на С

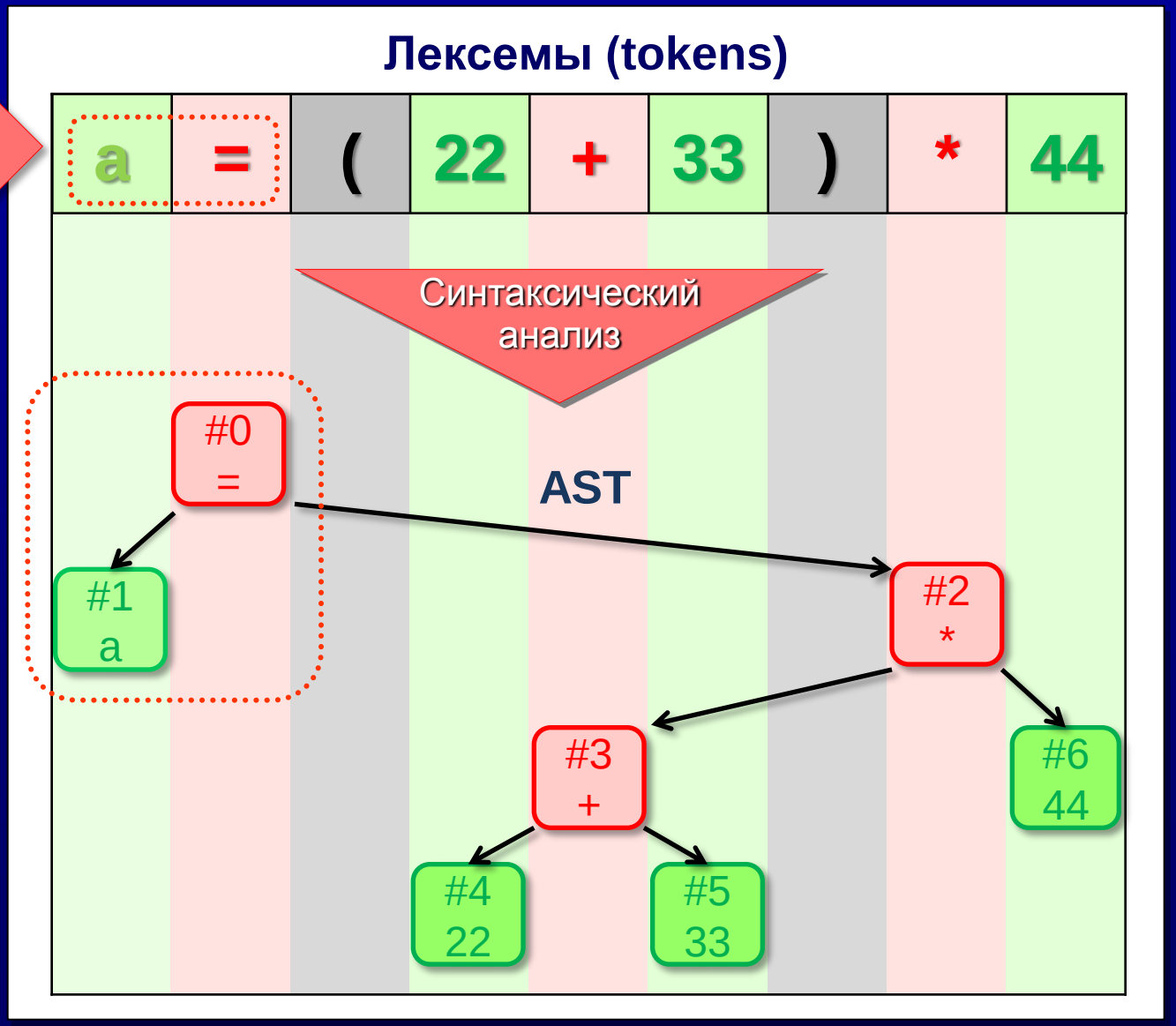
Лексический анализ

```
a = (22 + 33) * 44;
a =  22 + 33  * 44;
```

Пример оператора присваивания

Слева – лист с именем переменной.

Справа – поддерево со значением (выражением)



ТРАНСЛЯЦИЯ ЯЗЫКОВ ВЫСОКОГО УРОВНЯ

Текст на С

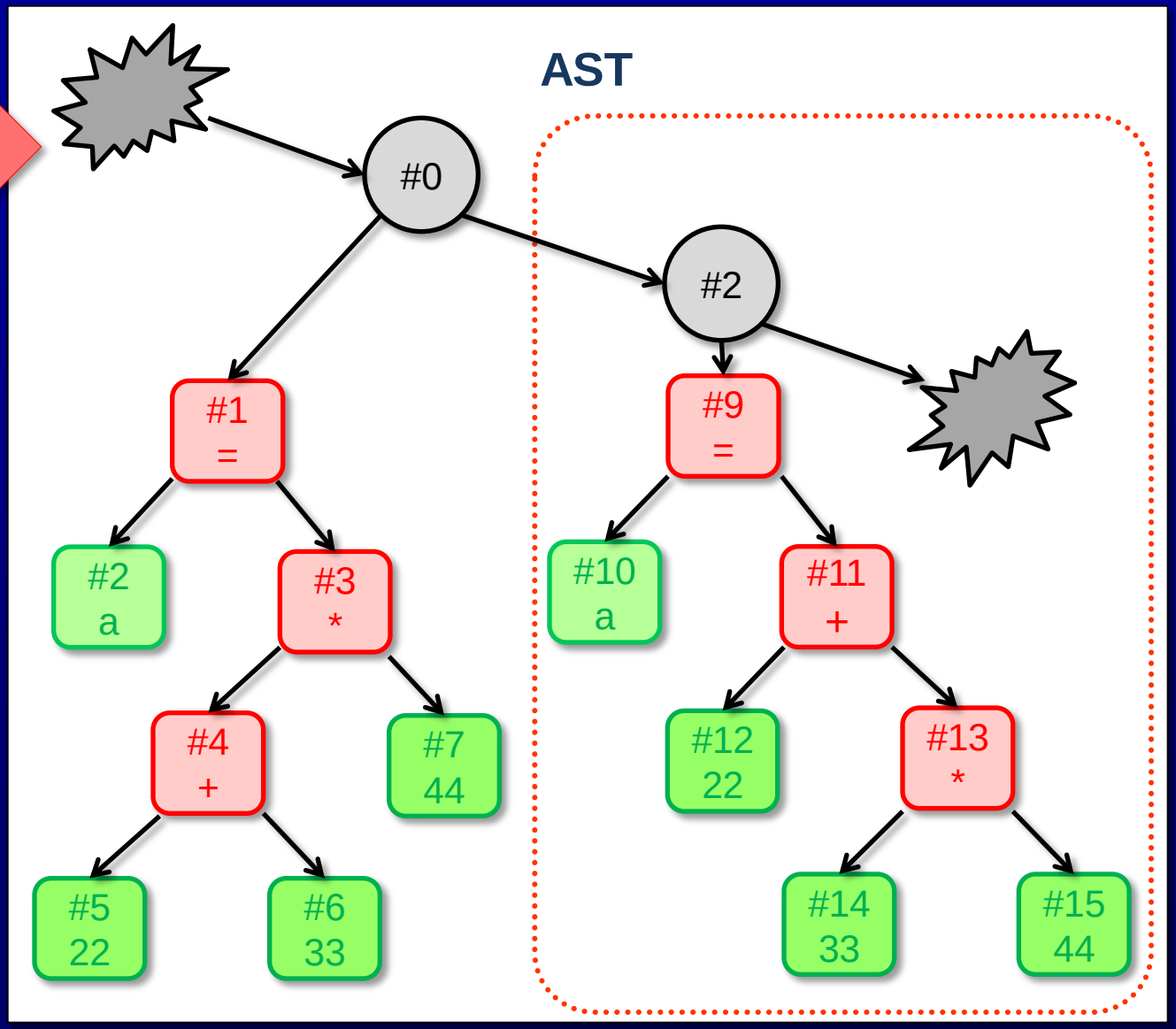
Лексический анализ

```
a = (22 + 33) * 44;
a = 22 + 33 * 44;
```

Пример оператора объединения команд

С обеих сторон прикрепляются команды.

Первая команда – слева.



ТРАНСЛЯЦИЯ ЯЗЫКОВ ВЫСОКОГО УРОВНЯ

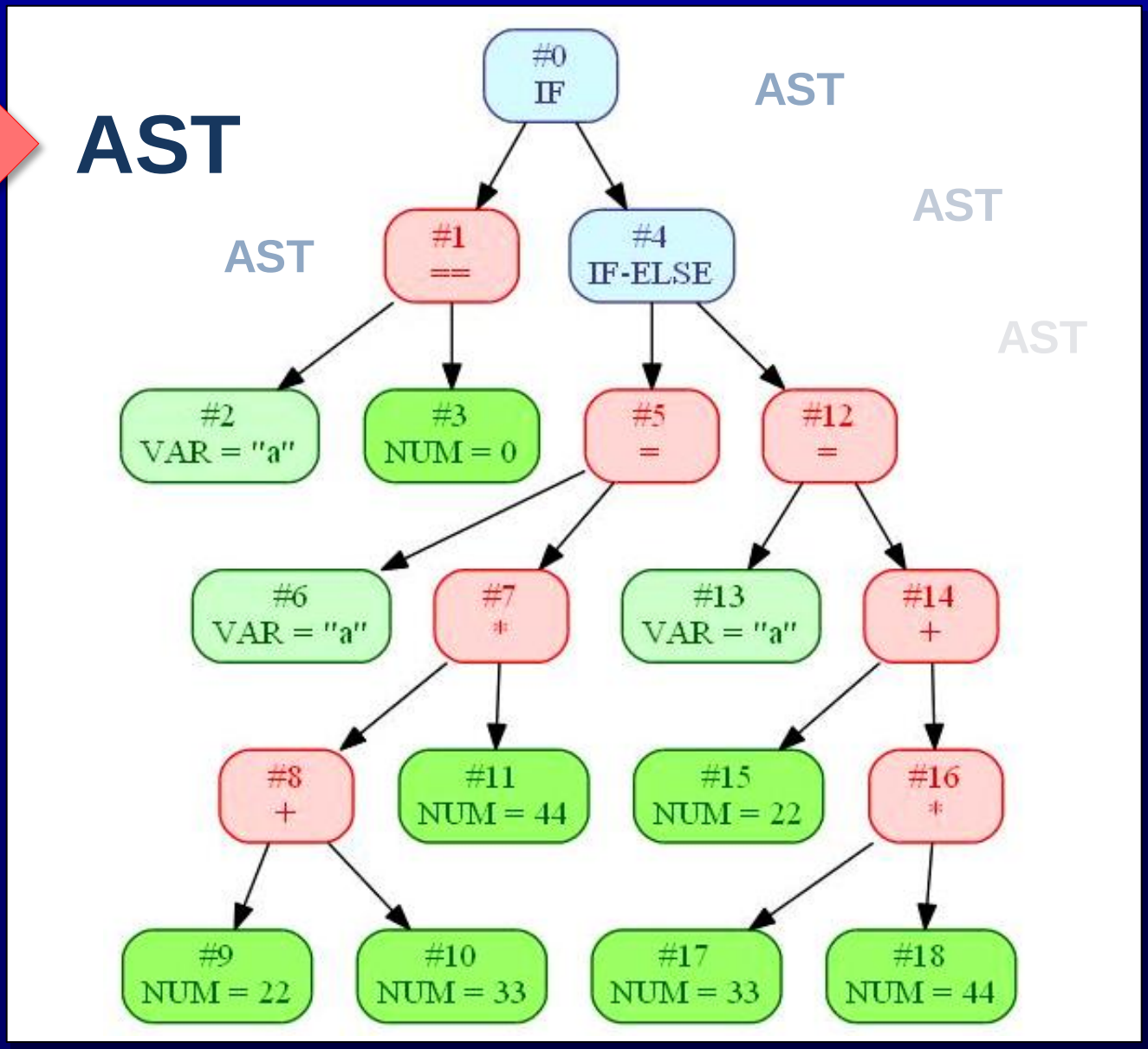
Текст на С

Лексический анализ

```
if (a == 0)
{
  a = (22+33) * 44;
}
else
{
  a = 22 + 33*44;
}
```

Пример программы большего размера.

Показана реализация различных команд и конструкций



ГРАФ ТРАНСЛЯЦИЙ ЯЗЫКОВ

SCC_C

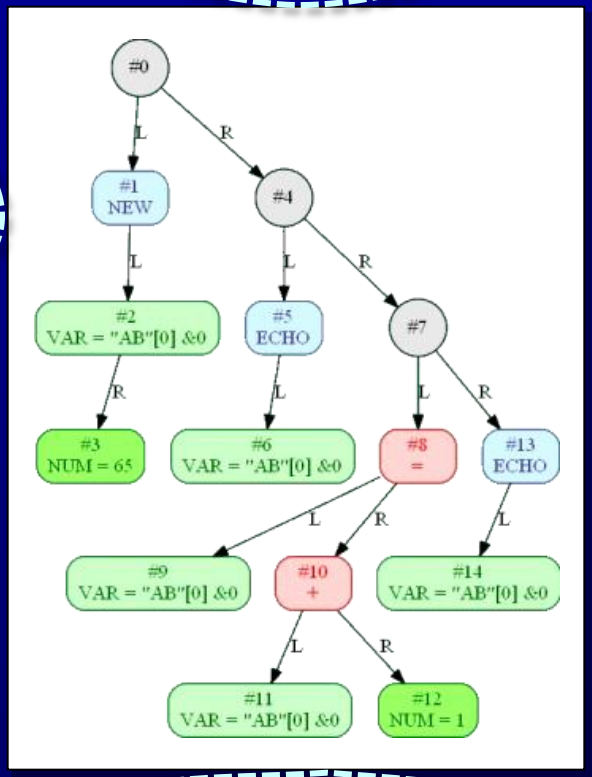
```
new AB = 'A';
echo AB;
AB = AB + 1;
echo AB;
```

C ↔ … → …

C ↔ Shakespeare

C → Asm

… C → …



SCC_Asm

```
push  65
popm  1
pushm 1
@
pushm 1
push  1
sum
popm  1
pushm 1
@
```

Shakespeare → Asm

SCC_Shakespeare

```
"A Simple Shakespeare program"

Dramatic Personae

Achilles, harum-scarum man.
Agrippa,  var #0 keeper.

Act I.
Scene I.

[Enter Agrippa and Achilles]

Achilles:
   You are as dirty as the sum of
   blue horrid charming fat
   distasteful fair door and rose.
Agrippa:
   You Agrippa.
   Speak your mind.
Achilles:
   You the sum of Agrippa and hero.
Agrippa:
   You are as vile as Agrippa.
   Speak your mind.

[Exeunt]
```

ТРАНСЛЯЦИЯ В SHAKESPEARE

Текст на SCC_C

Трансляция

```
new a;
a = 8;
```

Константы в Shakespeare

- Задаются *хорошими* и *плохими* словами из словаря
- Каждое словарное прилагательное увеличивает константу вдвое

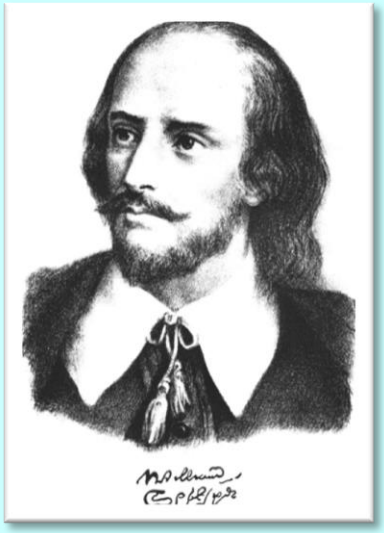
Текст на Shakespeare

"The simplest program"

Dramatic Personae

Achilles, harum-scarum man.

Agrippa, "a" place keeper.



Act I.

Scene I.

[Enter Agrippa and Achilles]

Achilles:

You handsome golden cunning embroidered pony.

2 * 2 * 2 * 2 * (+1) = 8

[Exeunt]

ТРАНСЛЯЦИЯ В SHAKESPEARE

Текст на SCC_C

Трансляция

```
new a;
a = (+1)+(-1)*(0);
```

Математические действия

По записи легко восстановить дерево:

<действие>
<левая часть>
and
<правая часть>

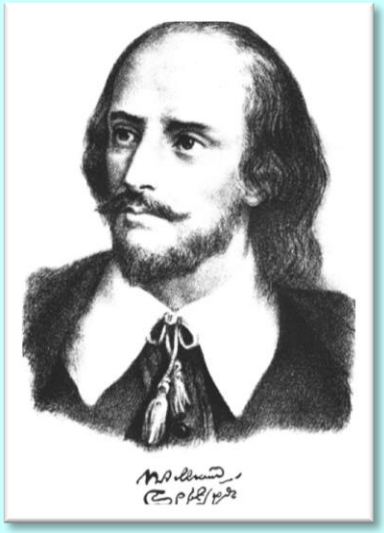
Текст на Shakespeare

"The simplest program"

Dramatic Personae

Achilles, harum-scarum man.

Agrippa, "a" place keeper.



Act I.

Scene I.

[Enter Agrippa and Achilles]

Achilles:

You the sum of Heaven and the product of hound and zero.

+	(+1)	*	(-1)	(0)

[Exeunt]

«ЛИТЕРАТУРНОСТЬ» SHAKESPEARE

Имитация литературных произведений:
необходимость «литературной красоты»

Способы «литературизации»

- **Создан эффективный и «литературно красивый» генератор констант.**

$42 = 32 + 8 + 2$ ⇒ 14 слов / 9 прилагательных
$42 = 6^2 + 6$ ⇒ 16 слов / 8 прилагательных

Меньше констант – текст «живее»

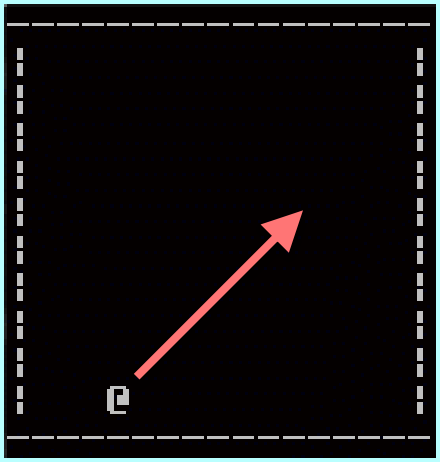
- **Разработан менеджер актеров на сцене.**
  Он позволяет оптимально выводить и уводить актеров, уменьшая количество ремарок («*очистить сцену*» и т. п.).

- **Язык был дополнен реализацией вызова функций, вписывающейся в общую концепцию Shakespeare.**

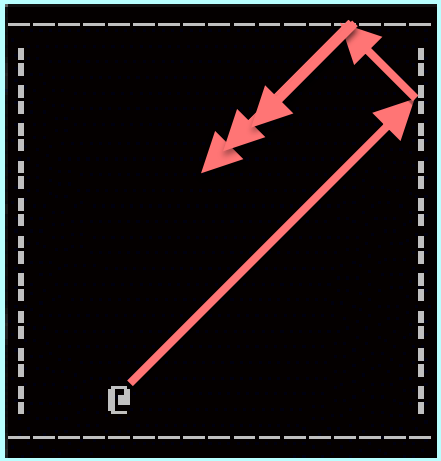
JIT-КОМПИЛЯТОР ДЛЯ INTEL x86

- Прямая замена командами процессора CPU Intel x86 (вместо вызова внутренних функций) более 70% команд
- Работа с математическим сопроцессором (FPU)
- Увеличение скорости в среднем в 2.5 - 3 раза по сравнению с Soft-процессором

Soft processor



JIT-processor



СТАТИСТИКА (МЕТРИКИ) ПРОЕКТА

- 65 файлов исходных текстов на C++ и ассемблере
- Более 17 000 строк кода
- Репозиторий кода: github://discharged-spider/SCC

```
switch (Data.Descriptor)
{
    case (N_NODE):
    {
        dtNodeStyle ().shape ("circle");

        dtNodeStyle ().fontcolor ("black")
                      .color     ("black")
                      .fillcolor ("#DDDDDD");

        break;
    }

    case (N_NUM):
    {
        dtNodeStyle ().fontcolor ("darkgreen")
                      .color     ("darkgreen")
                      .fillcolor ("#98FF66");

        n -= _snprintf (Label + MaxSize - n, n

        break;
    }
```

```
void DebugOutComand ()
{
    int Code;
    double* ESP;

    __asm __volatile__
    (
        "mov %%ebx, %0 \n"
        "mov %%esp, %1 \n"
        : "=m"(Code), "=m"(ESP)
        :
        : "ebx", "esp"
    );

    ESP += 8;

    static int BASE_ESP = -1;
    if (BASE_ESP == -1) BASE_ESP = (int)ESP + 16;

    int ESPSize = (BASE_ESP - (int)ESP) / 8;

    printf ("[============]\n");
    printf ("ESP = %d\n", ESPSize);

    printf ("Comand to execute = ");
    #define COMAND_DEF(NUM, NAME, DESCRIPTOR, PARA
    #include "..//Syntax//ComandDef.h"
    #undef COMAND_DEF
```

РЕЗУЛЬТАТЫ

Разработаны

- Высокоуровневый C-подобный скриптовый язык.
- Модификация языка программирования «Шекспир» с поддержкой вызова функций.
- Формат AST для единого внутреннего представления программ.
- Модульная расширяемая архитектура системы с возможностью двунаправленной трансляции.

Реализованы

- Транслятор с модульной архитектурой для трех языков.
- Оптимизатор для AST.
- Независимая система визуализации AST.
- Софт-процессор для исполнения программ.
- Вариант софт-процессора с возможностью JIT-компиляции в машинные коды семейства процессоров Intel x86.

СПАСИБО ЗА ВНИМАНИЕ

Автор проекта выражает благодарность
М. Д. Смолину