# Функции в языке StudentVM
# Вариант реализации

Петухов Дмитрий Сергеевич
171 группа, мат-мех
СПбГУ
2013

# Введение

Функции в языках программирования - это довольно мощный инструмент, позволяющий программисту выполнять определённый участок кода неограниченное число раз. Они позволяют не только сократить объём кода, но и повысить читабельность программы, поэтому крайне важно добавить возможность их использования в язык StudentVM[1].

# Идея реализации

Допустим, в программе произошёл вызов функции F. Учитывая то, что в SVM использована идея гарвардской архитектуры, необходимо выбрать, как и где именно будет происходить обработка вызова функции.

Предлагается следующее: Помимо основного стека - MainFrame (MF), - для каждой функции после её вызова создавать отдельный стек - StackFrame (SF), - на который помещаются:

1. Адрес возврата (по сути номер интерпретируемой строки, на которой произошёл вызов, InstructionPointer). При этом он не доступен для изменений (реализуется внутри интерпретатора).
2. Аргументы передаваемые в функцию. (тут стоит отметить, что было бы удобнее, если их количество обговаривается при объявлении функции, а не в момент её вызова).

Заводить стек под каждый вызов удобнее, чем организовывать всё это непосредственно в памяти, к тому же данный вариант позволяет использовать локальные “переменные”, которые будут заводиться в SF.

# Семантика и синтаксис

Предлагается следующий вариант декларации функций:

```
fun: foo
     rna 3
     [some calculations]
     ret
```

Объясним, что именно здесь происходит.

После ключевого слова “fun:” указывается имя функции. Затем указывается количество аргументов необходимых для нормальной работы подпрограммы, это делается с помощью использования ключевого слова “`rna`” - required number of arguments - каждый вызов функции будет требовать именно это количество чисел на MF.

[1] Далее SVM

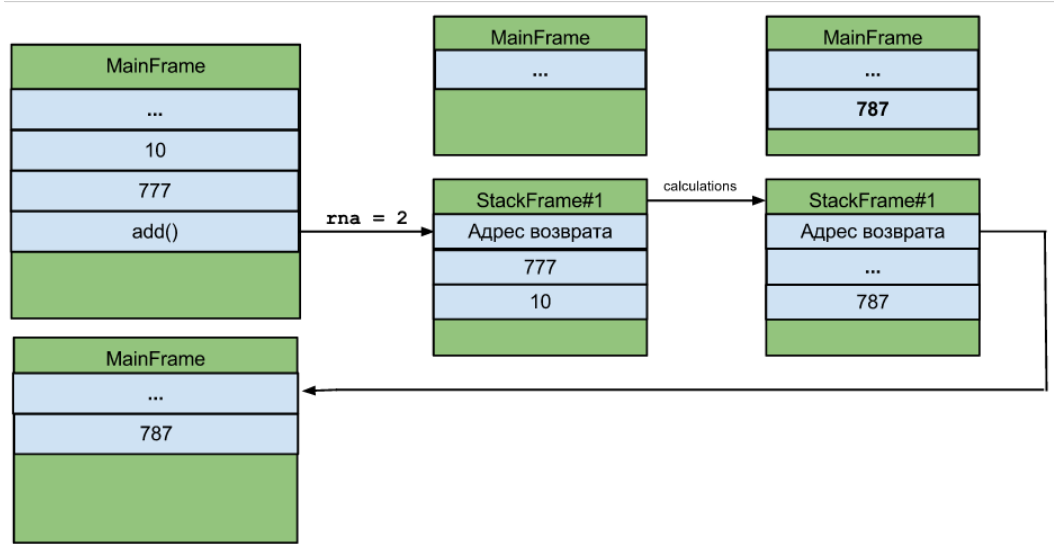
Далее выполняются определённые программистом вычисления. Заканчивается описание функции ключевым словом “ret”, которое возвращает на MF последнее число с вершины SF, и заканчивает выполнение подпрограммы путём перехода по адресу возврата.

Для вызова функции  предлагается следующий синтаксис:

```
...
;вызов функции с именем f
cll f
...
```

Приведём пример, поясняющий процесс вызова функции:

```
fun: myAdd
     rna 2
     add
     ret
     ...
     ldc 10
     ldc 777
     cll myAdd
     ...
```




Следует отметить, что данный вариант реализации позволяет использовать функции рекурсивно, ведь ничто не мешает завести ST2 для ST1, ST3 для ST2 и т. д.