Карпов В.Э.

Объектно-ориентированное программирование

Смолток. Лекция 4.

Байт-код

Байт-код

- **Байт-код** (*byte-code*) - это машинно-независимый код низкого уровня.
- Байт-код занимает промежуточное положение между результирующим объектным (исполняемым) кодом и интерпретируемой программой.
- Для выполнения инструкций байт-кода требуется наличие специальной программы – **интерпретатора**, называемого также виртуальной машиной.

Байт-код

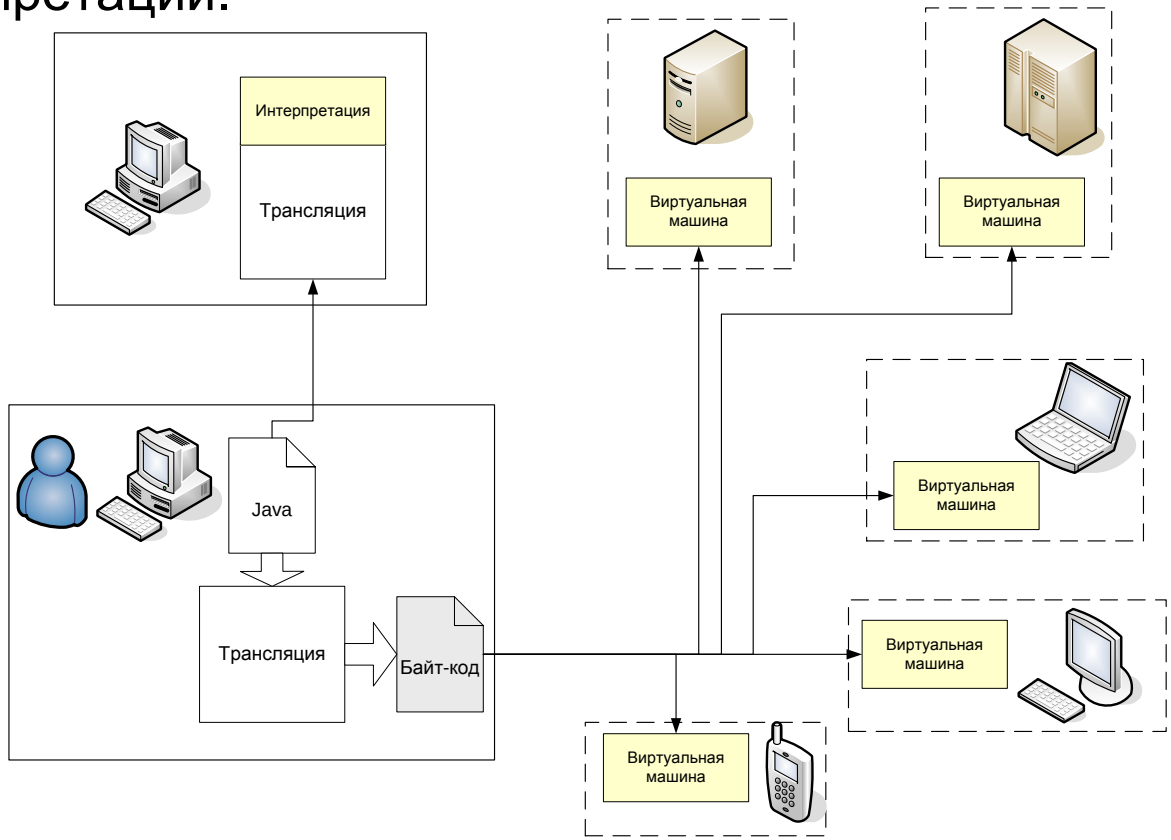
- Переносимость.Один и тот же байт-код может исполняться на разных *платформах* и *архитектурах*.
- Эффективность. Поскольку байт-код обычно менее абстрактный, более компактный и более приближен к машинному уровню, чем исходный код, эффективность байт-кода обычно выше, чем чистая интерпретация исходного кода.
- Многие интерпретируемые языки на самом деле транслируют исходный текст в байт-код и далее запускают интерпретатор байт-кода (Perl, PHP, Ruby и Python).
- Программы на Java обычно передаются на целевую машину в виде байт-кода, который перед исполнением транслируется в машинный код «на лету».

Простота и переносимость

Основное преимущество байт-кода - легкая переносимость.

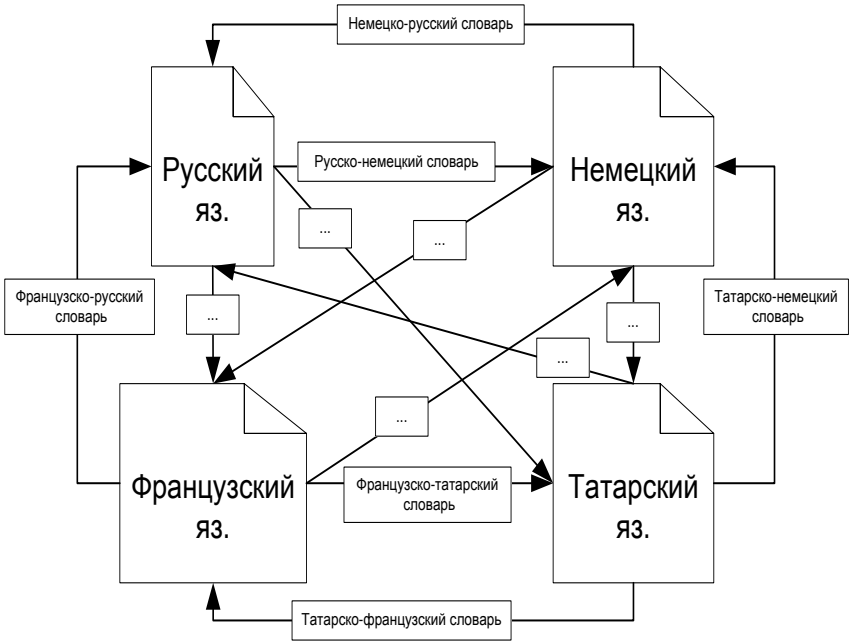
Большая часть работы по обработке исходного текста программы выполняется транслятором, создающим байт-код, то на ВМ приходится лишь интерпретация этого «почти машинного» кода.

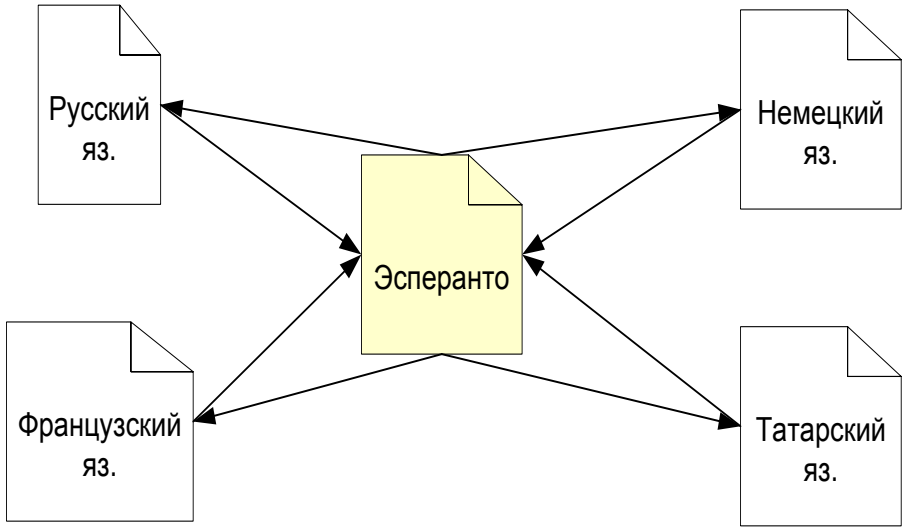
Фактически – это разбиение на отдельные этапы трансляции и интерпретации.



Б-К – язык-посредник

- 4 языка: 12 словарей.
- Если есть язык-посредник, то 8 словарей.
- Если N языков, то без посредника надо иметь N*(N-1) словарей (почти N^2), а с посредником – N*2.



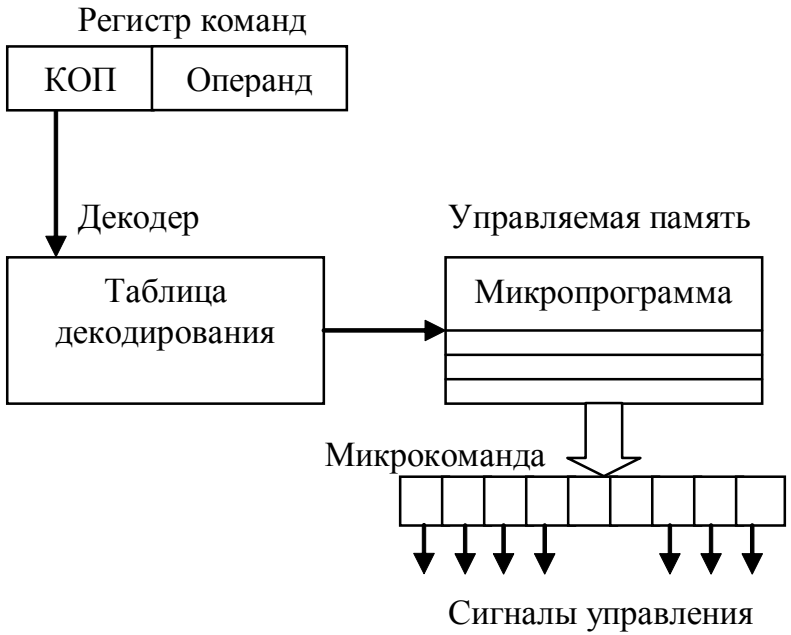


Простота Б-К

- Простота байт-кода позволяет реализовывать его интерпретаторы на уровне микропрограммного обеспечения процессоров. Благодаря этому существуют специальные Java-, Forth- и проч. машины (процессоры).
- Большинство инструкций байт-кода эквивалентны одной или нескольким машинным командам (командам ассемблера).
- Название «байт-код»: длина каждого кода операции — один байт. Более того, в байт-коде имеется тенденция реализовывать все команды по возможности в одном байте (при высокой частоте выполнения команд), что позволяет создавать компактный объектный код. Разумеется, вся команда может занимать более одного байта (код операции от 0 до 255, за которым следуют такие параметры, как регистры или адреса памяти).

История Б-К

- Концепция аппаратно-независимого исполняемого кода появилась еще в начале 1970-х годов.
- Разработки Н.Вирта по созданию виртуальной машины для языка Паскаль, а также Л.Петера Дойча в связи с созданием Лисп-машины.
- Тогда речь шла больше не о программной переносимости, а об идее персонального компьютера как специализированной машины с аппаратно реализованным языком высокого уровня.



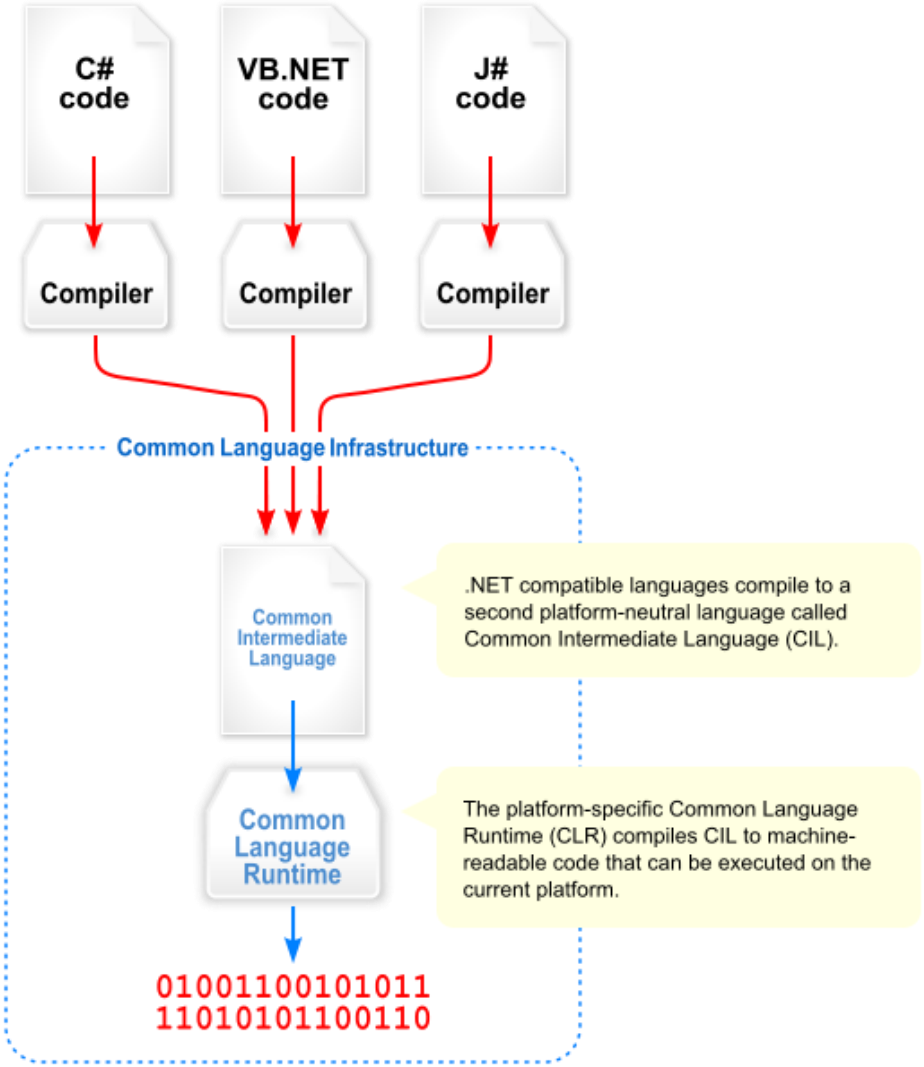
•Виртуальная машина.
•Байт-код.
В байт-коде имеется тенденция реализовывать все команды по возможности в одном байте (при высокой частоте выполнения команд), что позволяет создавать компактный объектный код.

Байт-код и спецпроцессоры

- Возможно создание процессоров, для которых данный байт-код является непосредственно машинным кодом (пример - процессоры для Java и Forth).
- **p-код** (**p-code**). Похож на байт-код, но физически может быть менее лаконичным и сильно варьироваться по длине инструкции. Он работает на очень высоком уровне, например: *«напечатать строку»* или *«очистить экран»*. P-код используется в СУБД и некоторых реализациях BASIC и Паскаля.

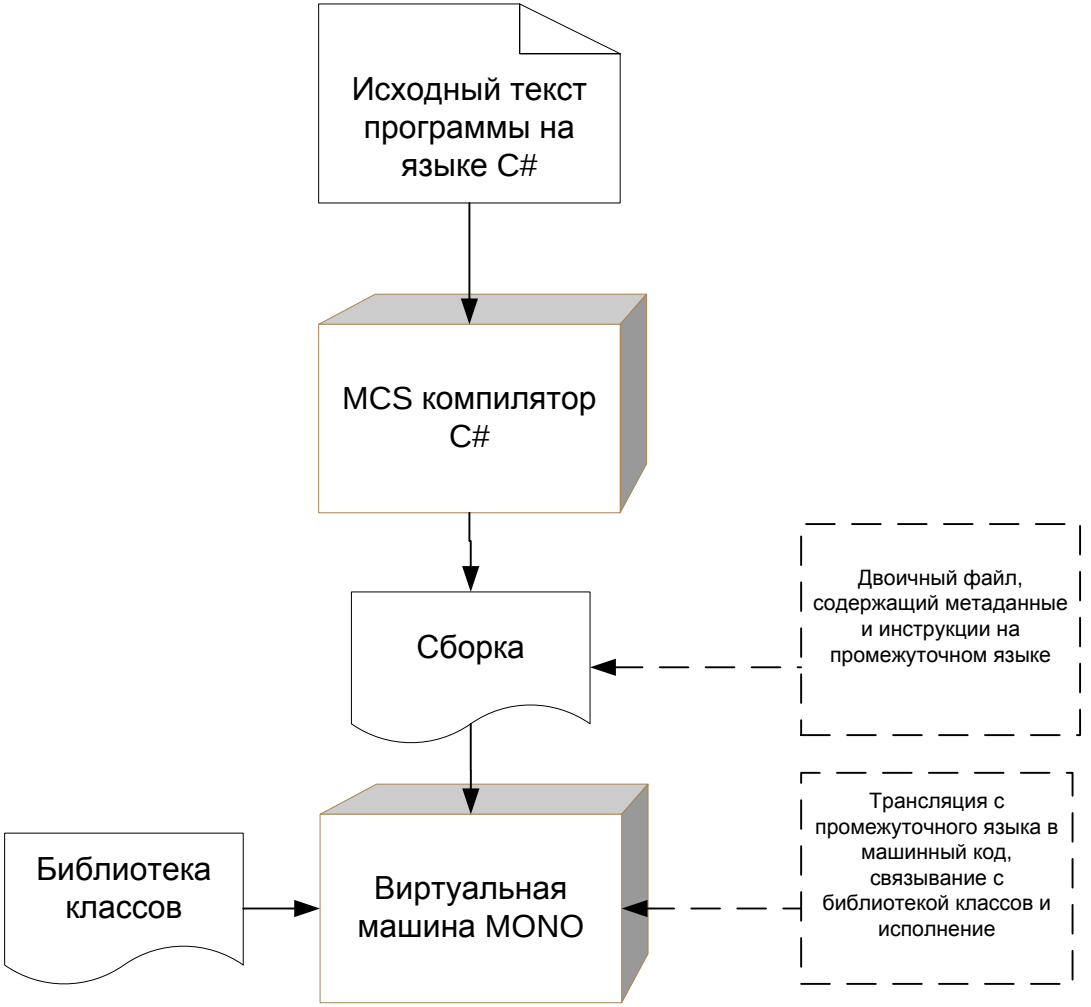
NET Framework

- Основа платформы - общеязыковая среда исполнения Common Language Runtime (CLR)
- Программа для .NET Framework сначала переводится компилятором в единый для .NET промежуточный байт-код **Common Intermediate Language (CIL).** В терминах .NET получается *сборка*, англ. *assembly*.
- Затем код либо исполняется виртуальной машиной **Common Language Runtime (CLR)**



MONO

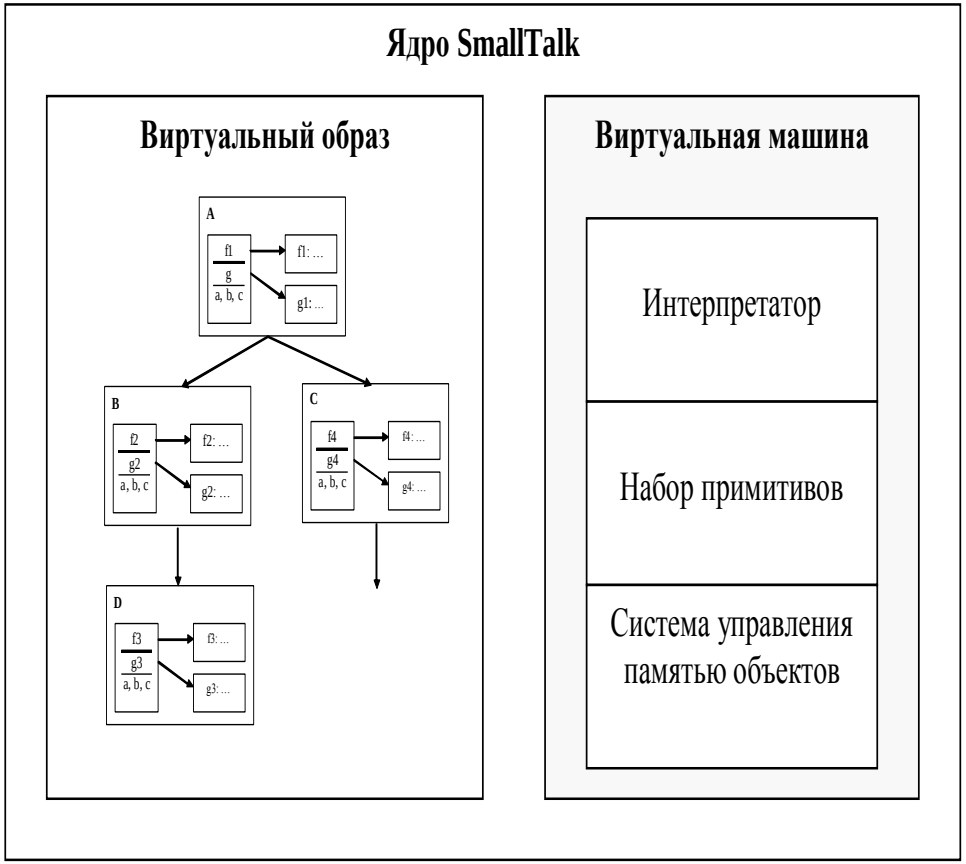
- Проект по созданию полноценного воплощения системы .NET Framework на базе свободного программного обеспечения. Основной разработчик - компания Xamarin (Novell).
- Mono - официальная реализация .NET на Unix-подобных операционных системах (Linux, Mac OS X и др.).



Ядро Смолток и примитивы

- Язык реализации системы и входной язык программирования - Смолток. На нем написана вся **машинно-независимая** часть, составляющая около 85% всей системы (виртуальный образ).

- **Машинно-зависимая** (виртуальная машина) часть реализована на низком уровне, в т.ч. и микропрограммно. При переносе системы на другую машину эта часть переписывается полностью

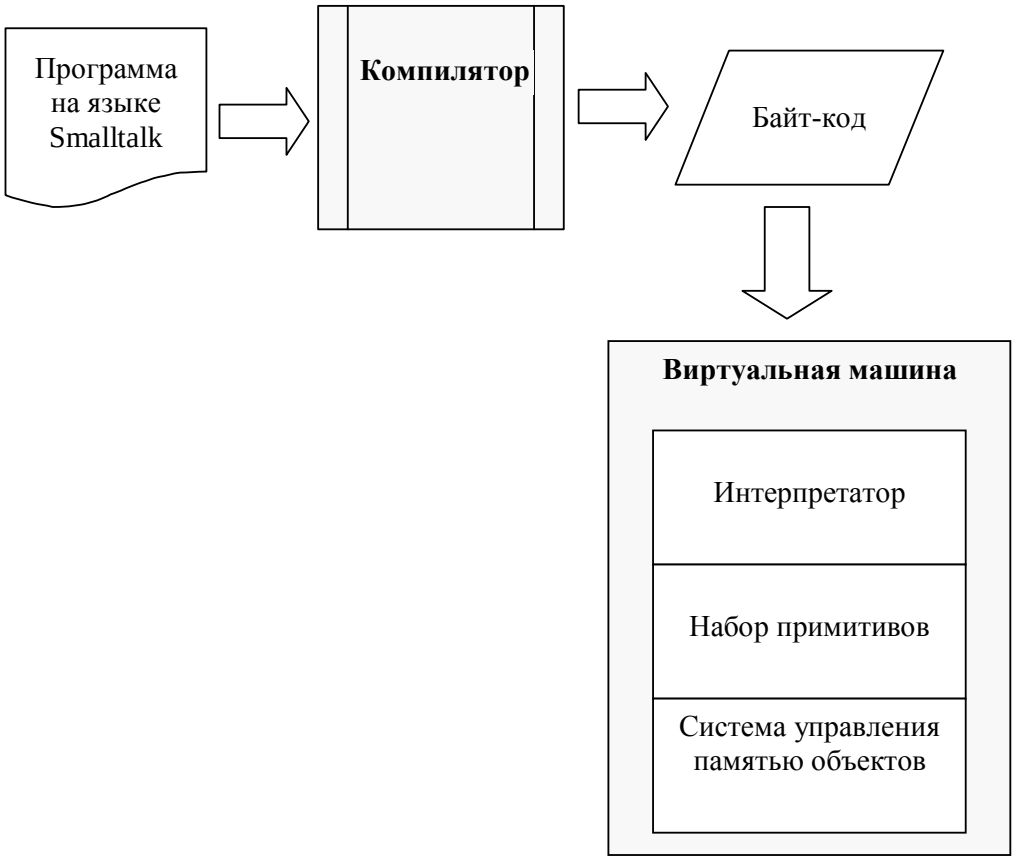


Виртуальный образ - откомпилированный образ написанной на Смолтоке части системы. Это встроенный набор классов и методов (за исключением примитивных методов).

256 примитивов

Виртуальная Смолток-машина

- Компилятор переводит программу с языка Смолток в промежуточный код стековой виртуальной машины (байт-код)
- Скомпилированные методы в виде байт-кодов интерпретируются ВМ.
- ВМ состоит из трех частей: интерпретатора, набора примитивных методов и системы управления памятью объектов.



Функции виртуальной машины

- **Функция интерпретации.** Интерпретатор считывает команды языка (байт-код) и выполняет их.
- **Управление объектами.** Блок управления объектами создает необходимые объекты и передает их интерпретатору, а ставшие ненужными объекты собирает и использует для дальнейшей работы.
- **Система базовых операций.** Операции ввода-вывода, управления процессами и другие базовые операции. В системе также регистрируются в качестве элементарных методов (primitive method - примитивы) те операции, которые нельзя реализовать на самом Смолтоке (или их реализация неэффективна) и которые реализуются в виде программ вне системы Смолток.

Контекст метода

При посылке сообщения создается объект, называемый контекстом метода. В него входит следующая информация:

1. контекст вызова;
2. адрес команды;
3. указатель стека;
4. обрабатывающая процедура метода (объекты с байт-программами);
5. получатель;
6. аргументы;
7. временные переменные;
8. стек.

Эта информация необходима для выполнения метода.

Примеры групп команд виртуальной Смолток-машины

1) Проталкивание в стек переменных экземпляра-получателя. Переменные экземпляра фиксируются для каждого экземпляра, и в каждом объекте для них отводится область памяти. Данная команда проталкивает в стек считанные переменные экземпляра, в частности – получателя.
   Байт-коды 0-15, 128:
   0-15 [0000iiii] Помещение в стек переменной-экземпляра получателя с номером #iiii.
   128 [10000000] [jjkkkkkk] Помещение в стек переменной экземпляра получателя, временной переменной, литерала, глобальной переменной, указываемой литералом [jj] с номером #kkkkkk.
2) Проталкивание в стек временной переменной. Временные переменные создаются в момент вызова метода.
   Байт-коды 16-31, 128:
   16-31 [0001iiii] Помещение в стек временной переменной с номером #iiii.
3) Проталкивание символов в стек. Символ - это селектор сообщения или константа с объектным указателем.
   Байт-коды 32-63, 128:
   32-63 [001iiiii] Помещение в стек литерала с номером #iiiii.
4) Вызов метода с использованием селектора сообщения, находящегося в области литералов. Команда производит поиск селектора сообщения, начиная со словаря класса получателя. Если поиск успешен, то производится вызов соответствующего метода.
   Байт-коды 131, 132, 134, 208-255.

Примеры групп команд. Продолжение

5. Помещение в стек активного контекста. Команда помещает в поля текущего контекста значения регистров и затем помещает в стек указатель этого контекста.
   Байт-код 137.
6. Команды перехода и условного перехода.
   Байт-коды 144-175:

144-151 [10010iii] Переход по адресу iii+1.
152-159 [10011iii] Выталкивание из стека, переход по адресу iii+1 при значении false вытолкнутой вершины.
160-167 [10100iii] [jjjjjjjj] Переход по адресу (iii-4)*256+jjjjjjjj.

7. Посылка заявки на вычисление. Команда реализует арифметические операции "+" и "-". Если получатель не является целым числом, то выполняются действия, аналогичные обычной посылке заявки.

Байт-коды 192-207:
192-207 [1100iiii] Посылка специальной заявки #iiii.