Автор: Дмитрий Андреевич Тимофеев
Отредактировал: Александр Сергеевич Герасимов, alexander.s.gerasimov@ya.ru

КОМАНДЫ ВИРТУАЛЬНОЙ СТЕКОВОЙ МАШИНЫ МИЛАНА

Виртуальная машина Милана содержит память команд, память данных и стек.
В памяти команд находятся исполняемые инструкции, или команды.
Память данных используется для хранения значений переменных.
Стек является рабочей областью: команды виртуальной машины считывают
свои аргументы из стека и заталкивают в стек результаты.

Виртуальная машина Милана исполняет следующие команды.

NOP

Отсутствие операции; команда пропускается.

STOP

Прекращает выполнение программы; виртуальная машина возвращает
управление операционной системе.

LOAD <адрес>

Помещает в стек слово, расположенное в памяти данных по адресу <адрес>.

STORE <адрес>

Выталкивает из стека слово и записывает его в память данных по адресу
<адрес>.

BLOAD <адрес>

Помещает в стек слово, адрес которого вычисляется следующим образом:
<адрес в памяти данных> = <адрес> + <значение с вершины стека>.
Значение с вершины стека удаляется.

Команду BLOAD можно использовать, например, для обращения к элементу
массива или к переменной, адрес которой вычисляется во время
выполнения программы.

BSTORE <адрес>

Вычисляет адрес в памяти данных по формуле:
<адрес в памяти данных> = <адрес> + <значение с вершины стека>,
значение с вершины стека при этом удаляется. Выталкивает из стека
слово и записывает его по адресу <адрес в памяти>.

Команду BSTORE можно использовать для записи значения в элемент
массива или в переменную, адрес которой вычисляется во время
выполнения программы.

Пусть стек имеет вид: [10, 20, ...]. В этом случае команда BSTORE 5
запишет по адресу 15 число 20.

PUSH <значение>

Заталкивает <значение> в стек.

POP

Выталкивает слово из стека.

DUP

Заталкивает в стек значение, равное значению на вершине стека.

ADD

Выталкивает из стека два числа и заталкивает в стек результат их сложения.

MULT

Выталкивает из стека два числа и заталкивает в стек результат их умножения.

SUB

Выталкивает из стека число <a>, выталкивает из стека число <b> и заталкивает в стек разность <b> - <a>.

Пример: после выполнения последовательности команд

```
PUSH 10
PUSH 8
SUB
```

на вершине стека будет находиться число 2.

DIV

Выталкивает из стека число <a>, выталкивает из стека число <b> и заталкивает в стек частное <b> / <a>. Используется целочисленное деление. Если значение <a> = 0, то диагностируется ошибка времени исполнения.

INVERT

Меняет знак числа на вершине стека на противоположный.

COMPARE <код>

Выталкивает из стека число <a>, выталкивает из стека число <b>, затем заталкивает в стек результат сравнения <b> и <a>.

Операция сравнения определяется значением аргумента <код>:

<код>	операция
0	<b> = <a>
1	<b> != <a>
2	<b> < <a>
3	<b> > <a>
4	<b> <= <a>
5	<b> >= <a>

Результатом сравнения является значение 1, если указанное соотношение выполняется, и 0 в противном случае.

Пример: после выполнения последовательности команд

```
PUSH 5
PUSH 7
COMPARE 2
```

на вершине стека будет находиться значение 1 (так как 5 < 7).

JUMP <адрес>

Выполняет переход к команде по адресу <адрес> в памяти команд, то есть устанавливает адрес следующей исполняемой команды равным <адрес>.

JUMP_YES <адрес>

Выполняет переход к команде по адресу <адрес> в памяти команд, если на вершине стека находится ненулевое значение; слово выталкивается из стека.

JUMP_NO <адрес>

Выполняет переход к команде по адресу <адрес> в памяти команд, если на вершине стека находится значение 0; слово выталкивается из стека.

INPUT

Считывает целочисленное значение со стандартного устройства ввода и заталкивает его в стек. В случае ошибки ввода-вывода или несоответствия формата диагностируется ошибка времени исполнения.

PRINT

Выталкивает из стека слово и выводит его на стандартное устройство вывода. Это слово форматируется как целое число и завершается переходом на новую строку.

Каждая команда исполняется за один такт работы виртуальной машины.

Виртуальная машина формирует содержимое памяти команд и данных, читая и интерпретируя текстовый файл программы. Файл имеет следующий формат.

Файл читается как последовательность строк. Символ ';' начинает комментарий, все символы, начиная с него и до конца строки, игнорируются.

Команда состоит из адреса, кода операции и аргумента, если он требуется. Адрес является целым числом, за которым следует символ ':'. Нумерация команд начинается с 0.

Код операции совпадает с описанным ранее символьным обозначением, например: STOP. Если команда имеет аргумент, он должен быть целым числом.

Пример корректной программы на языке виртуальной машины Милана:

```
------------------------------------------------------------
0:      INPUT
1:      STORE   42              ; n := READ
2:      LOAD    14
3:      PUSH     4
4:      LOAD    42
5:      COMPARE  2
6:      JUMP_NO  9
7:      PUSH    10
8:      STORE   42
9:      LOAD    42
10:     PRINT
11:     STOP
------------------------------------------------------------
```

Для инициализации памяти данных используется служебная инструкция виртуальной машины SET:

```
        SET <адрес> <значение>
```

В результате исполнения инструкции SET по адресу <адрес> в память данных записывается слово <значение>.

Инструкция SET не сохраняется в памяти команд. Она служит для инициализации памяти перед началом исполнения программы. Компилятор может генерировать блок команд SET для задания значений констант и статически инициализированных переменных.

Инструкция SET <адрес> <значение> эквивалентна паре команд

```
        PUSH <значение>
        STORE <адрес>
```

и выполняется за один такт.

В инструкции SET перед кодом операции не указывается адрес.
Пример корректного использования SET:

```
-----------------------------------
SET     0       15
SET     1       40

0:      LOAD    0
1:      LOAD    1
2:      ADD
3:      PRINT
4:      STOP
-----------------------------------
```

В результате выполнения этой программы будет напечатано число 55.