**Заводов Андрей Павлович (P3119)**

**Спецификация виртуальной машины**

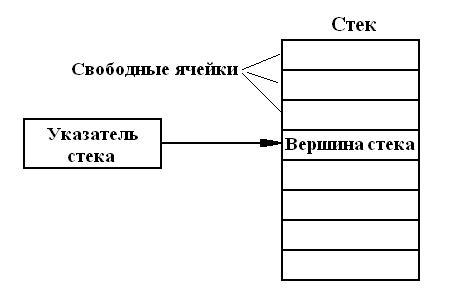
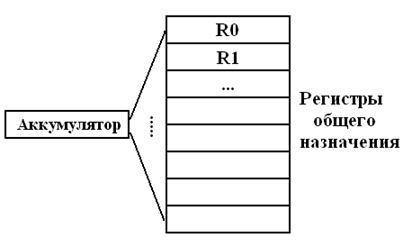
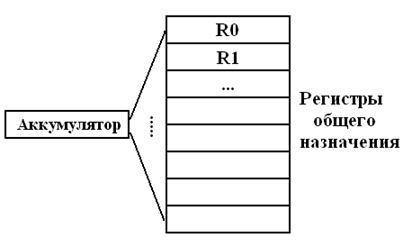
**«Edem-VM»**

**1. Общие сведения**

**Edem-VM** – простейшая виртуальная машина, предназначенная для работы с целыми числами и символами. Предусматривает операторы ветвления, циклы и массивы размерностью до 256 полей.

**2. Модель вычислений**

Данная VM может использовать как стековую модель вычислений, так регистровую. Инструкции для работы с ними приведены ниже.



**Регистры специального**

**назначения**

**…**

**…**

**R512**

**R257**

**R256**

**3. Модель памяти**

Данные и промежуточные результаты будут опционально хранится либо в стеке (размер стека - 512 полей), либо в одном из 512 регистров, доступных для хранения данных, причем первые 256 регистров специального назначения для функций, доступ к этим регистрам не ограничен, но перед вызовом и после завершения функции они обнуляются. Из данного утверждения следует, что при вызове из функции другой функции, данные старой функции не сохранятся (для их сохранения, следует поместить данные на стек).

|  |  |
| --- | --- |
| **Номера регистров** | **Назначение регистров** |
| **0** | Регистр для хранения вершины стека. |
| **1-256** | Регистры доступные для функций, исполняемых в данный момент. После завершения функции, данные регистры обнуляются. |
| **257-511** | Регистры общего назначения. |

\*По умолчанию значения всех регистров равны 0

**4. Набор инструкций**

|  |  |  |
| --- | --- | --- |
| **Название инструкции** | **Мнемоника** | **Описание инструкции** |
| **4.1. Для работы со стеком** | | |
| **Add** | **Add <vl>** | Добавляет на вершину стека **<vl>**. |
| **Print** | **Print** | Вывод верхнего элемента стека на экран. |
| **Pop** | **Pop <reg>** | Копирует верхний элемент стека в регистр **reg**. |
| **Sum** | **Sum** | Cложение двух верхних элементов и запись результата обратно в стек. |
| **Pow** | **Pow** | Умножение двух верхних элементов и запись результата обратно в стек. |
| **Div** | **Div** | Целочисленное деление второго элемента на первый и запись результата обратно в стек. |
| **Diff** | **Diff** | Вычитание из второго элемента первого и запись результата обратно в стек. |
| **4.2. Для работы с регистрами** | | |
| **R** | **R-<number>** | Обращается к значению регистра **<number>** |
| **RAdd** | **RAdd <reg> <vl>** | Записывает значение **value** в регистр по номеру **reg**. |
| **RPrint** | **RPrint <reg>** | Выводит на экран значение регистра по номеру **reg**. |
| **RSum** | **RSum <vl-1> <vl-2> <reg>** | Суммирует значения **<vl-1>, <vl-2>** и записывает результат в **<reg>**. |
| **RPow** | **RPow <vl-1> <vl-2> <reg>** | Умножает значение **<vl-1>, <vl-2>** и записывает результат в **<reg>**. |
| **RDiv** | **RDiv <vl-1> <vl-2> <reg>** | Осуществляет целочисленное деление **<vl-1>** на **<vl-2>** и записывает результат в **<reg>**. |
| **RDiff** | **RDiff <vl-1> <vl-2> <reg>** | Вычитает значение **<vl-2>** из **<vl-1>** и записывает результат в **<reg>**. |
| **4.3. Организация ветвлений** | | |
| **If, else** | **IF <**(**condition)>**  **{Operators-true}**  **{Operators-false}** | Выше представлена схема ветвлений. Если **condition=true** выполняется инструкция на следующей стоке за условием. Если **condition=false** выполняется инструкция через строку от условия. |
| **4.4. Организация циклов** | | |
| **While** | **While <(condition)> {**  **Operator1**  **Operator2}** | Операторы в фигурных скобках (Operator1, Operator2) выполняются до тех пор, пока условие **<(condition)>,** не станет ложным |
| **4.5. Безусловный переход** | | |
| **goto** | **label-<number>**  **goto-<number>** | Оператор безусловного перехода. Переводит (**goto**) интерпретатор на строку с определенной меткой(**label**). |
| **4.6. Специальные символы** | | |
| **Коммент.** | **//** | Строка после этого символа не читается компилятором. |
| **#** | **#<value>** | Распознавать **value**, как код символа. |
| **4.7. Работа с файлами** | | |
| **AlgRead** | **AlgRead <path>** | Читает алгоритм из файла по пути **path** |
| **ValWrite** | **ValWrite <path> <reg>** | Сохраняет значение из регистра **reg** по пути **path**. |
| **ValRead** | **ValRead <path> <reg>** | Читает файл по пути **path** и сохраняет первое корректное значение в регистр **reg**. |
| **StWrite** | **StWrite <path>** | Cохраняет верхнее значение стека в файл по пути **path**. |

**\*** Если у инструкции несколько аргументов, все они передаются через пробел.

**\*\*** Под **<vl>** подразумевается либо число, либо значение регистра (**R-<number>**),

**\*\*\*** Под **<reg>** подразумевается только значение регистра (**R-<number>**).

\*\*\*\* Под **<number>** подразумевается целое число, либо значение какого-либо регистра (конструкция вида – R-R-5 (обращение к регистру, номер которого находится в ячейки 5), считается корректной).

**5. Режим адресации**

**Непосредственная адресация –** данные для инструкции входят в саму инструкцию.

Загрузка и получение данных производится с помощью специальных инструкций.

Данные могут быть заданы как непосредственно в самом коде алгоритма, так и прочитаны из файла.

**6. Форма записи алгоритма**

Алгоритм подается на стандартный поток чтения либо загружается из файла целиком (не по 1 инструкции). Инструкции располагаются каждая на новой строке. Если компилятор обнаруживает ошибку в инструкции или инструкция ему не известна, исполнение алгоритма не завершается, но выводится сообщение об ошибки. Данная строка в свою очередь просто игнорируется исполнителем.

**7. Мнемоническое представление программ**

**7.1. Представление функций**

Каждое описание новой функции начинается с ключевого слова **func**, далее через дефис идет имя функции.

Аргументы для своей работы функция берет с вершины стека.

Далее на M строках, каждая из которых начинается с символа «\_» (подчеркивание) идут инструкции этой функции.

Результат своей работы, функция записывает на вершину стека, либо изменяет данные в зависимости от своей логики и назначения.

Для вызова функции необходимо указать уникальный идентификатор функции, перед этим поместив на стек необходимое количество параметров.

**Примечание**: не безопасно вызывать функцию, предварительно не поместив на стек необходимое количество аргументов. Если в стеке окажется меньше аргументов, чем требуется функции, программа завершится с ошибкой.

Пример:

**func-1**

\_**Pop** **R**-1

\_**Pop** **R**-2

\_**RAdd R**-1 **R**-2 **R**-3

\_**Add** **R**-3

\_**Print**

Функция сохраняет 2 значения со стека в специальные регистры 1 и 2, в регистр 3 записывает их сумму. Возвращает данную сумму на стек и выводит ее на экран.

**7.1. Представления алгоритма**

Каждая инструкция алгоритма записывается на новой строке. Для выделения блоков инструкций используются “{” – для начала блока и ”}” – для окончания.