ScribbleVM Specification

The Scribble blah blah

Registers

ScribbleVM is a register based virtual machine. Registers are fixed size (64 bit) areas in memory used to store values. In the case of ScribbleVM a large number of registers (Up to 255) are used. Most of these registers are used to store the values of variables however there are also several registers reserved for standard operations.

Register 0 is the program counter, it points to the location of the current instruction being executed.

Register 1 is the stack base pointer, it points to the location the stack was at when a function started execution (Note: This functionality is entirely up to the developer or compiler. It will only be set of somebody executes move $2 $1 at the start of a function and ensures it is pushed and popped when a function is called)

Register 2 is the stack current pointer, it points to the current location in the stack

Registers 3, 4 and 5 are temporary registers. They are reserved to be used by the compiler as storage locations for the current operation (Note: Again this is entirely up to the programmer or compiler, any registers may be used as temporary registers just as registers 3, 4 and 5 may be used for other purposes)

Primitives

The primitives in ScribbleVM are byte, short, int, long which are 1, 2, 4 and 8 bytes wide accordingly. When in registers they are all seen as 8 bytes wide (So the opcodes do not have to differentiate between primitive type before executing) and these sizes only apply to values within heap memory or in the constant region (Both the heap and constant region are elaborated on below).

Instructions

Instructions within ScribbleVM will all be 64 bits long. The first 8 bits of every instruction will represent the opcode while the remaining 7 bytes can be used to provide arguments for the opcode. In the case that an

instruction does not need 64 bits the remaining bits will be left empty but still included in the generated bytecode; the reason for this being that fixed size instructions make flow control simpler.

Instruction Set

The instruction set in ScribbleVM is split up into two regions ,the instructions an array of 64 bit instructions which ScribbleVM can execute and the constant region an area of memory used by instructions that need to use more than 7 bytes of data for arguments or use a varying amount of data.

Valid Instructions

Load – The load instruction is expressed as Load(consantAddress, register). The first 32 bits after the opcode represent the address in the constant area where the value to load can be found. The next 8 bits represent the destination register (Where the value should be placed).

```
Load {
        Address : 32 bits integer,
        Destination: 8 bits unsigned byte
}
```

A load operation may point to a data element such as a string which should be placed on the heap or it may be a primitive (See above).

The result of a load operation will vary by type, in the case of a primitive element it is placed directly into a register. In the case of a heap item an entry for it will be created on the heap and the heap address will be placed in the destination register.

//TODO: Formalize what is expected at the constant address. Make sure to address usability, speed and size. Be clear about what should be there.

Move – The move instruction is expressed as Move(from, to). The first 8 bits after the opcode represent the register who's value should be taken. The next 8 bits represent the destination register (Where the value should be placed).

```
Move {
        Address : 8 bits unsigned byte,
        Destination: 8 bits unsigned byte
}
```

A move operation takes whatever in the from register and places it in the to register. For example the sequence

load 1 $3
move $3 $4

would result in the the values of registers 3 and 4 both being equal to 1.