VVM Program example 1

A simple VVM Assembly Language program which adds an input value to the constant value -1 is shown below (note that lines starting with "//" and characters to the right of program statements are considered comments, and are ignored by the VVM machine).

```
// A sample VVM Assembly program
// to add a number to the value -1.
IN        Input number to be added
ADD 99    Add value stored at address 99 to input
OUT       Output result
HLT       Halt (program ends here)
*99       Next value loaded at address 99
DAT -001  Data value
```

This same program could be written in VVM Machine Language format as follows:

```
// The Machine Language version
901  Input number to be added
199  Add value stored at address 99 to input
902  Output result
000  Halt (program ends here)
*99  Next value loaded at address 99
-001 Data value
```

VVM Program example 2

```
// Example of simple conditional
// structure.
// Equivalent to the following BASIC
// program:
//   INPUT A
//   INPUT B
//   IF A >= B THEN
//      C = A + B
//   ELSE
//      C = A - B
//   ENDIF
//   PRINT C
//   END
in     Input A
sto 98 Store A
in     Input B
sto 99 Store B
lda 98 Load value of A
sub 99 Subtract B from A
brp 11 If A >= B, branch to 11
// A is < B Find difference
lda 98 Load value of A
sub 99 Subtract value of B
sto 97 Store C
br  14 Jump to 14
lda 98 [11] Load A (A is >= B)
add 99 Add B
sta 97 Store C
out    [14] Print result
hlt    Done
```

VVM Program example 3

```
// Simple looping example.
// Equivalent to the following BASIC
// program:
//   INPUT A
//   DO WHILE A > 0
//      PRINT A
//      INPUT A
//   LOOP
//   END
in      Input A
sto 99  Store A
brp 04  [02] If A >= 0 then skip next
br  10  Jump out of loop (Value < 0)
brz 10  [04] If A = 0 jump out of loop
lda 99  Load value of A (don't need to)
out     Print A
in      Input new A
sto 99  Store new value of A
br  02  Jump to top of loop
hlt     [10] Done
```

VVM Program example 4

```
//  Sample program to print the
//  square of any integer in the
//  range 1 - 31. Greater value will
//  cause a data overflow (you can
//  try this). Smaller value will
//  cause endless loop (try this
//  too)! Hint: If many iterations (e.g.
//  input > 4), set speed to FAST!
in       Input value to be squared
sto 99   Store input at 99
lda 98   Load current sum (top of loop)
add 99   Add value to sum
sto 98   Store the sum
lda 97   Load current index
add 96   Add 1 to index
sto 97   Store new index value
sub 99   Subtract value from index
brz 11   Jump out if index = value
br  02   Do it again (bottom of loop)
lda 98   Done looping - load the sum
out      Display the result
hlt      Halt (end of program)
//  Data used by program follows
*96      Resume loading at address 96
dat 001  Constant for counting
dat 000  Initial index value
dat 000  Initial sum
```