sub1

// 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

add2nos

//input and add two numbers

IN get first number

STO 99 store it at 99

IN get second number

ADD 99 add number at 99 to it

OUT output the sum

HLT

simpIf

//if x+y==0 (i.e. x=-y) then output 0 else output 1

IN

STO 99 x

IN y

ADD 99 y+x

BRZ 06 branch is sum is 0

LDA 98 load 1

OUT [06]

HLT

\*98

DAT 001

diff2nos

//[positive] difference between two numbers

IN

STO 98 x

IN

STO 99 y

SUB 98 y-x

BRP 08 branch if y>=x

LDA 98 else: x>y get x

SUB 99 x-y

OUT 08:

HLT

\*98

DAT 000 x

DAT 000 y

if1

// Example of simple conditional structure.

// 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

equals2nos

//test if two numbers equal

//1 if yes, 0 if no

IN get x

STO 99

IN get y

SUB 99 y-x

BRZ 07 branch if y==x

LDA 98 no: load 0

BR 08

LDA 97 yes: load 1

OUT over:

HLT

\*97

DAT 001

DAT 000

loop1

// Simple looping example.

// 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

loopsumming

//loop N times summing inputs

IN get N, the number of numbers to input

SUB 97 subtract 1 to avoid extra loop. kludge

STO 99

IN top of loop. get next number

ADD 98 add it with sum

STO 98 store new sum

LDA 99

SUB 97 subtract 1

STO 99

BRP 03 loop again if not yet 0

LDA 98 sum to output

OUT

HLT

\*97

DAT 001 constant 1

DAT 000 sum

square

// 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 x value to be squared

STO 99 Store x at 99

LDA 98 Load current sum (top of loop)

ADD 99 Add x to sum. sum=sum+x

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 x 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

//multiply x\*y

//loop y times adding x to itself

IN x

STO 99

IN get y, the number of times to loop

SUB 96 subtract 1 to avoid extra loop. kludge

STO 98 y is the loop counter, counting down

LDA 97 top of loop. load sum

ADD 99 add it with x

STO 97 store new sum

LDA 98 load loop counter y

SUB 96 subtract 1

STO 98 store y back

BRP 05 loop again if not yet 0

LDA 97 sum to output

OUT

HLT

\*96

DAT 001 constant 1

DAT 000 sum

DAT 000 y

DAT 000 x

problems:

x\*0

x\*-y

0\*y OK

-x\*y OK