```
NAS Instruction Set V0.4 (2015)
===============================

Changes:

Escape sequences can now be in strings and chars. E.g., "1st line\n2nd
line", '\t'. Only \n, \t, \" (for string), and \' (for char) have been
added. If you want other escapes, please add them to nas.l yourself.

A stack overflow check is added to nas.y. No checks added for stack
underflow which should never occur unless you change sp manually in your
code. There is also no check for max code size (which is now set to 3000);
max stack size is 5000; max # of labels is 1000.

NAS Instruction Set V0.2 (2011)
===============================

Descriptions that are obvious are omitted.

For preciseness, {} means "the content of" in the following.

The following push an item onto the stack, where N is an integer, R... is
a register: one of "sb", "fp", "ac", and "in" (also called "ac"); C is a
single-quoted character, and S is a double-quoted string.

Instruction     Description
  push N
  push C
  push R        pushes {R}  // the value contained in R
  push R[N]     pushes {{R} + N}}  // {R}+N is used as a stack address
  push R1[R2]   pushes {{R1} + {R2}}
  push S        pushes the "address" of S

  pop R
  pop R[N]
  pop R1[R2]

The following operate on the stack top, and are *destructive* (the
element(s) operated on are/is popped and replaced by the pushed result),
where T represents the top stack item and T2 the item below it. L is
a label.

  complt        T = if T2 < T then 1 else 0
  compgt
  compge
  comple
  compne
  compeq
  add
  sub           T = T2 - T
  mul
  div           T = T2 / T
  mod
  neg
  and
  or
  j1 L          jumps to L if T is 1 (T is popped)
  j0 L          jumps to L if T is 0 (T is popped)
  puti          prints T as integer with newline (T is popped)
  putc          prints T as char ...
  puts          prints T as string ...
  puti S        ... with format string
  puts S
  putc S
  puti_         ... without newline
  putc_
  puts_

These ones are not destructive:
```

```
   geti          reads and pushes an integer
   getc
   gets

   jmp L         unconditional jump

   call L, N     N >= 0 is the number of arguments
   ret           T = return value

   end           ends execution, or by end-of-file

   // ...        comment, from // to end-of-line

   ;             treated as white-space (can be used to separate
                 instructions on the same line)
```

Notes:

The stack is an integer stack. When dealing with characters, their ASCII codes (integers) are pushed. No packing/unpacking possible.

Strings are a primitive type, and are stored in some heap behind the scene. They are represented and operated on by their addresses (integers). They are "immutable". If you want mutable strings, build your own using chars.

Both string and char can be empty; push '' pushes the null char (0).

True and false are represented by integers 1 and 0 respectively

Labels are of the form Lnnn, where nnn = 000..999.

In the implementation (nas.y), sp is pointing at the next empty slot above top of the stack.

To "declare" an array say of size 100, you can fiddle with the stack pointer: push sp; push 100; add; pop sp.

Function call/return: please refer to the tutorial.

*** The NAS implementation (nas.l + nas.y) has virtually no error checking which should not be a problem since the assembly language input is coming from a compiler and not a human programmer. ***