Project parser example

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit
end
```

Symbol table: empty

String buffer: empty

Instruction buffer:

- Our goal in the parser is to read a program such as shown on the left, and turn it into code that can be executed on the VM.
- The parser has several tasks
  1. Decide what statement is being parsed (there is one per line)
  2. Add declarations of variables and labels to a *symbol table* which is used in processing variables and jumps
     1. For jumps, determine where in the program the jump goes to
     2. For variables, determine where in storage the variable is

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit
end
```

Symbol table: empty

String buffer: empty

Instruction buffer:

- The presence of functions (subroutines) in our code complicates life
- A function or subroutine is started by a *gosublabel label* statement and ended by a *return* statement
- A function will only contain a single *return*
- Variables declared in a function are
  - Only visible in the function (there scope is the function)
  - At runtime they will be put on the *data memory*, but we'll talk about that later
- Labels declared in a function are only visible on the function

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit
end
```

Symbol table: empty

String buffer: empty

Instruction buffer:

- Handling functions/subroutines will be the most complicated part of the program
- Strings are tracked using a *string buffer*
- Generated code is placed into *instruction buffer* to be written to the output file when the entire program has been processed
  - Statements cannot be written immediately because, for example, a jump may be to a label that hasn't been declared yet (see, e.g., L2 to the left)
  - Thus code for the jump cannot be finalized until the target of the jump is known.
- Variables are always declared before being used.

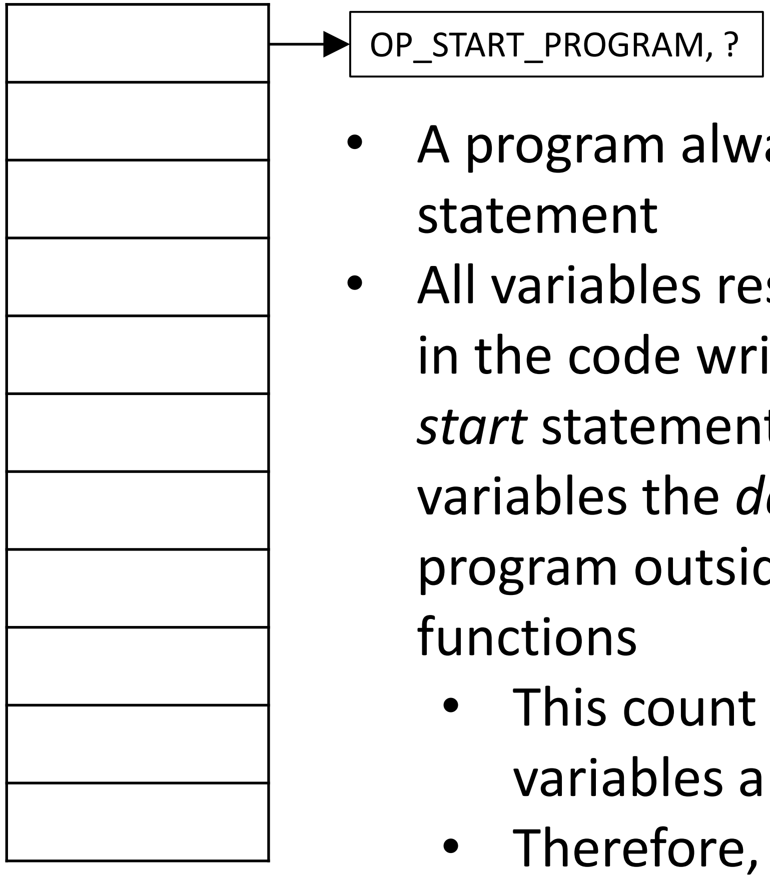
- The *instruction buffer* contains pointers to *Stmt* objects. A program always starts with a *start* statement.
- *Stmt* is an *abstract class* (discussed later this week) that all statement types inherit from.

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table: empty

String buffer: empty

Instruction buffer:



- A program always starts with a *start* statement
- All variables reside on the *data memory*, and in the code written to the output file for the *start* statement must contain the number of variables the *data memory* will hold for the program outside of those needed for functions
  - This count is not known until all variables are declared
  - Therefore, we leave that field in the OP_START_PROGRAM statement undefined

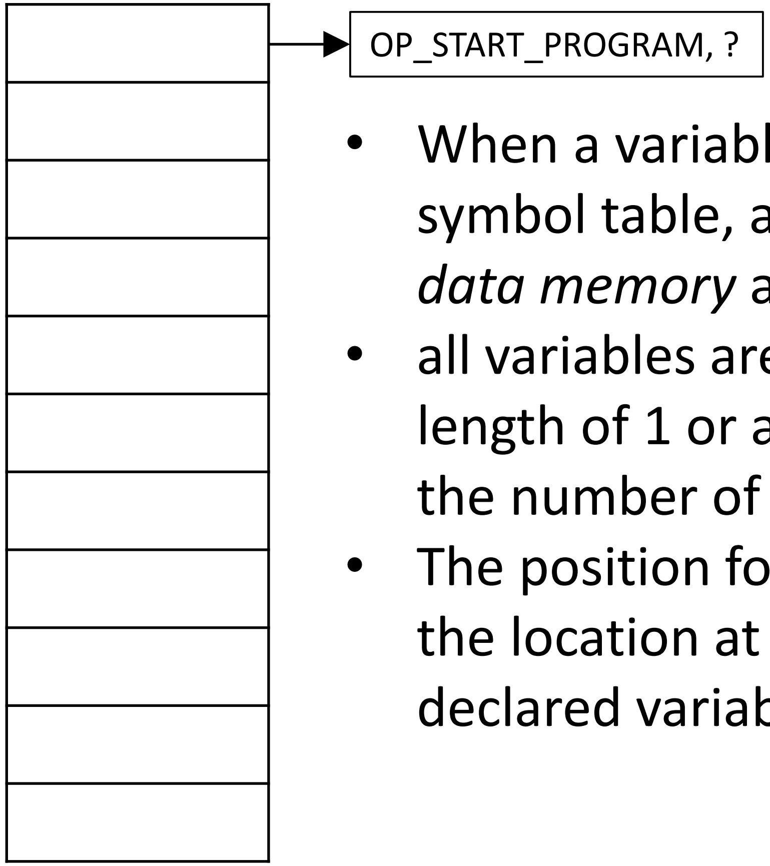
The *start* statement causes a Start statement object to be created and pointed to by the instruction buffer. The operand will be the size of data memory needed to hold outer scope variables, a value that is not yet known.

start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end

Symbol table: **A, <0,1>**

String buffer: empty

Instruction buffer:



- When a variable is declared, it is put into the symbol table, along with its position in the *data memory* and its length
- all variables are either scalar ints with a length of 1 or an array with a length that is the number of elements in the array
- The position for some variable is one plus the location at the end of the previously declared variable

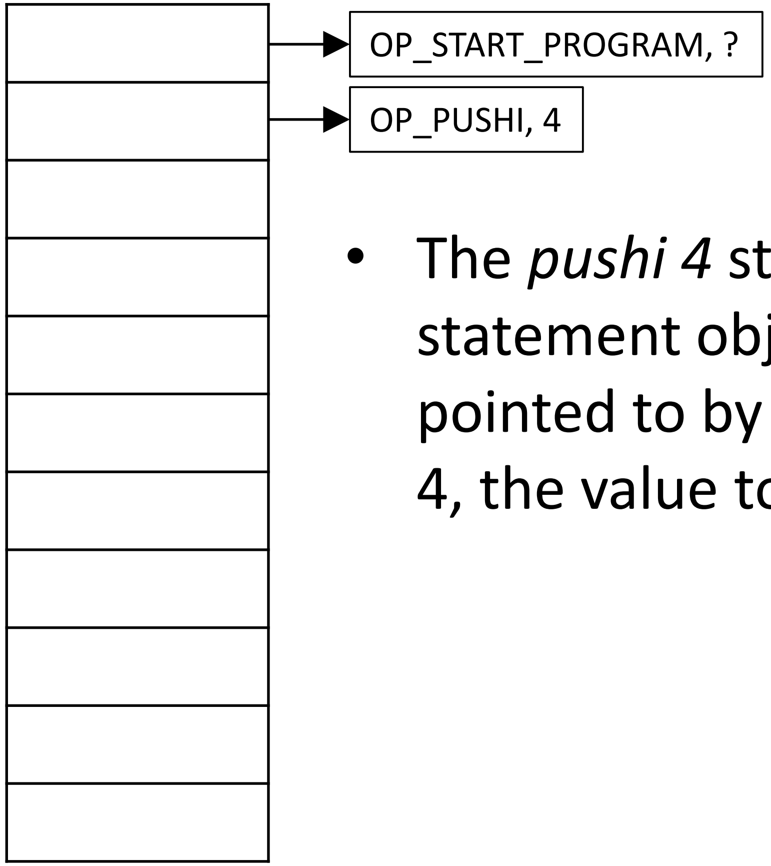
The *declscal A* statement causes and entry for A to be created in the current symbol table. Since this is the first declared variable, its location will be 0.

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table: **A, <0,1>**

String buffer: empty

statement buffer:



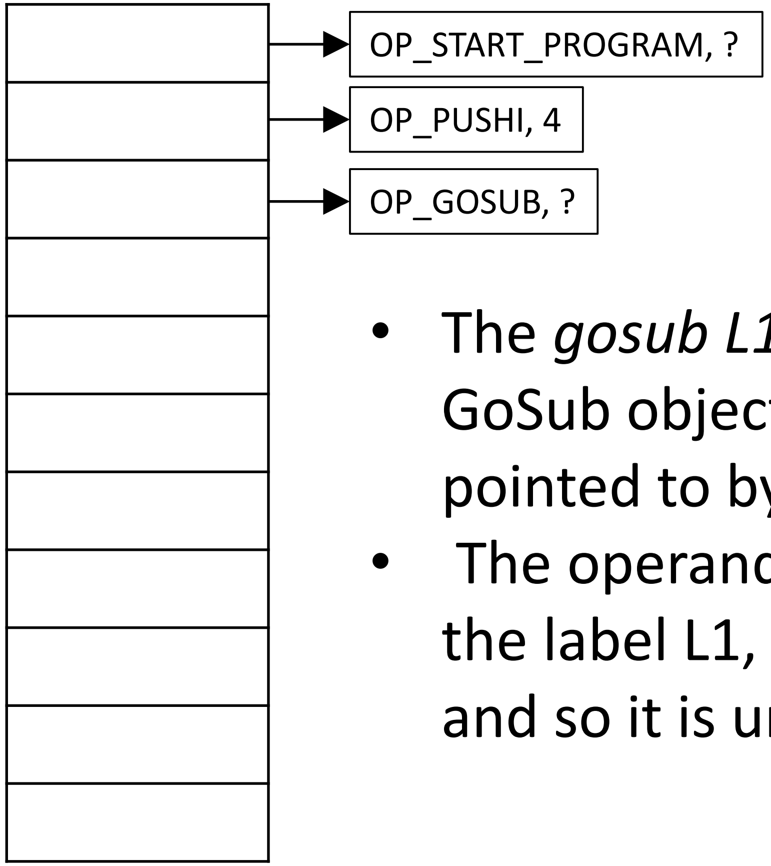
- The *pushi 4* statement causes a Push statement object to be created and pointed to by the buffer. The operand is 4, the value to be pushed.

start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end

Symbol table: **A, <0,1>**

String buffer: empty

statement buffer:



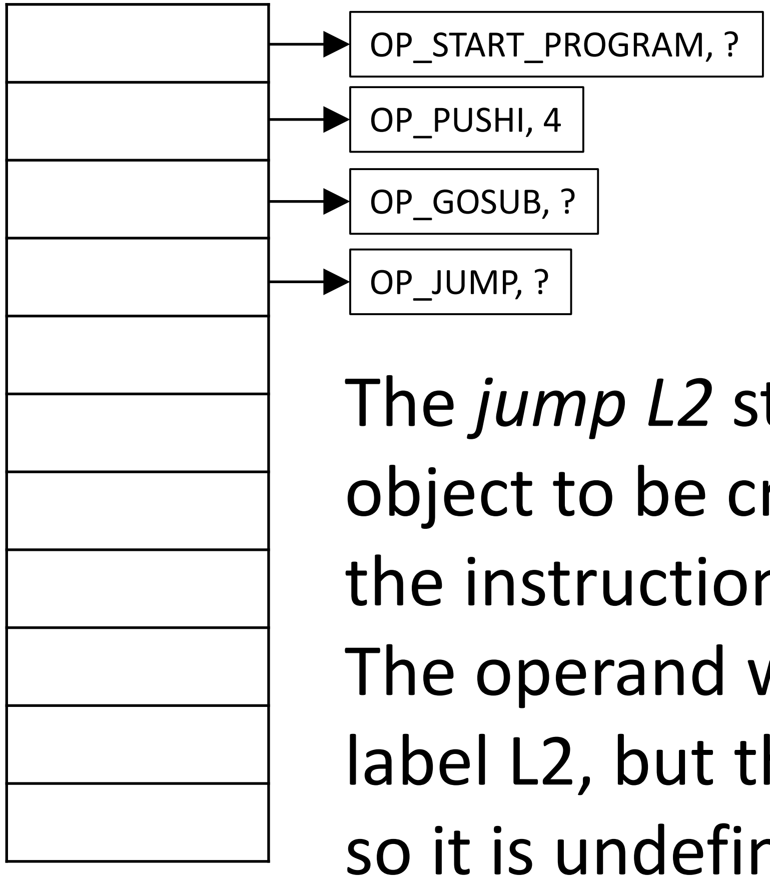
- The *gosub L1* statement causes an GoSub object to be created and pointed to by the instruction buffer.
- The operand will be the location of the label L1, but that is not yet known and so it is undefined for now.

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table: **A, <0,1>**

String buffer: empty

statement buffer:



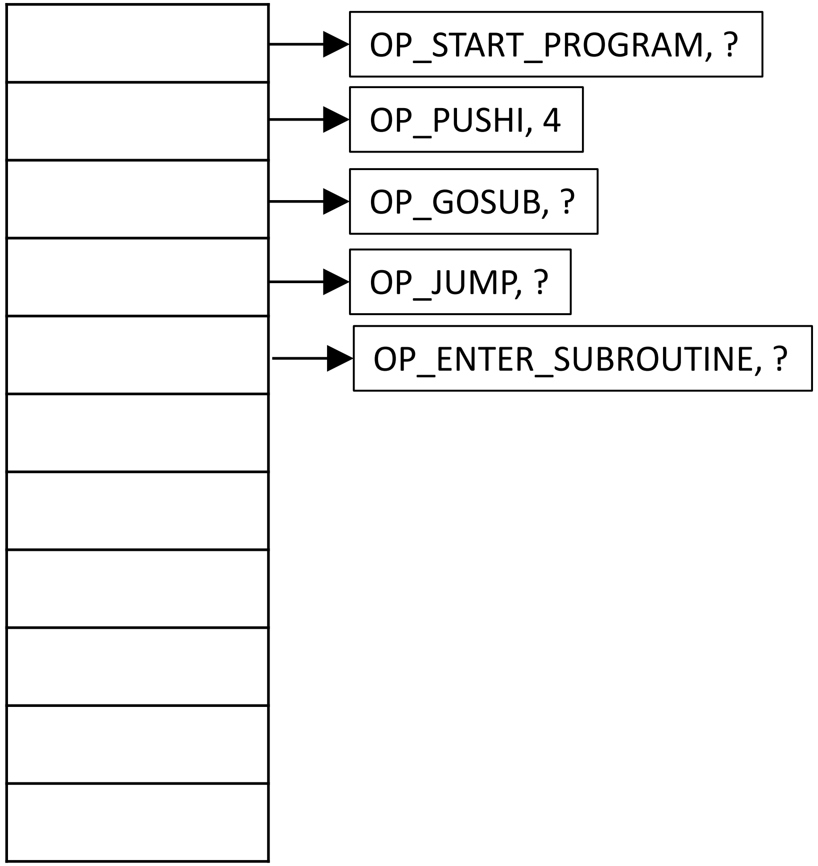
The *jump L2* statement causes an Jump object to be created and pointed to by the instruction buffer.
The operand will be the location of the label L2, but that is not yet known and so it is undefined for now.

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table: **A, <0,1>** | **L1 <4, 0>**

String buffer: empty

statement buffer:



- The *gosublabel L1* statement causes an GoSubLabel object to be created and pointed to by the instruction buffer.
- The label L1 is added to the outer scope symbol table, along with its location (4) and its length (zero) since it doesn’t occupy any space in the data memory.
- The operand will be the size of the *data memory* needed for the subroutine, which is not yet known, and so is undefined.

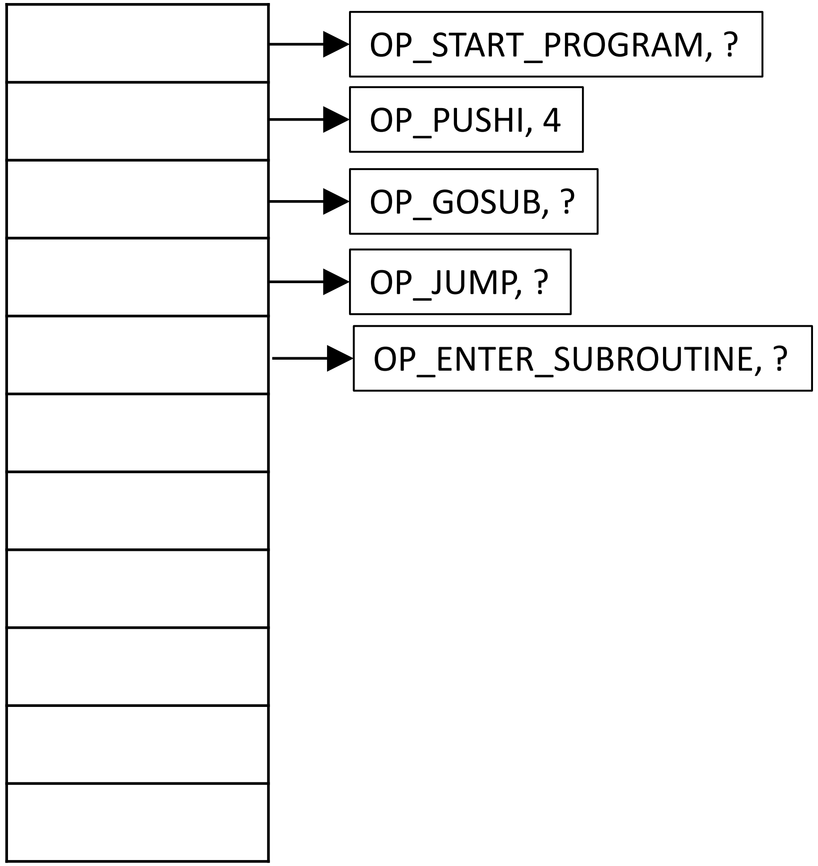
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**A <1, 0>**

String buffer: empty

statement buffer:



- The *declscal A* statement causes an entry for A in the symbol table.
- Since we are in a new scope (the subroutine) it is ok for there to be a name that is the same as a name in the outer scope.
- Since it is a scalar the length is 0.
- It is the first variable declared in the subroutine, so its location will be 1 plus the location of the last storage declared in the outer scope.
- We will get rid of the subroutine declarations at the end of the subroutine, so we need to mark where the subroutine table begins.

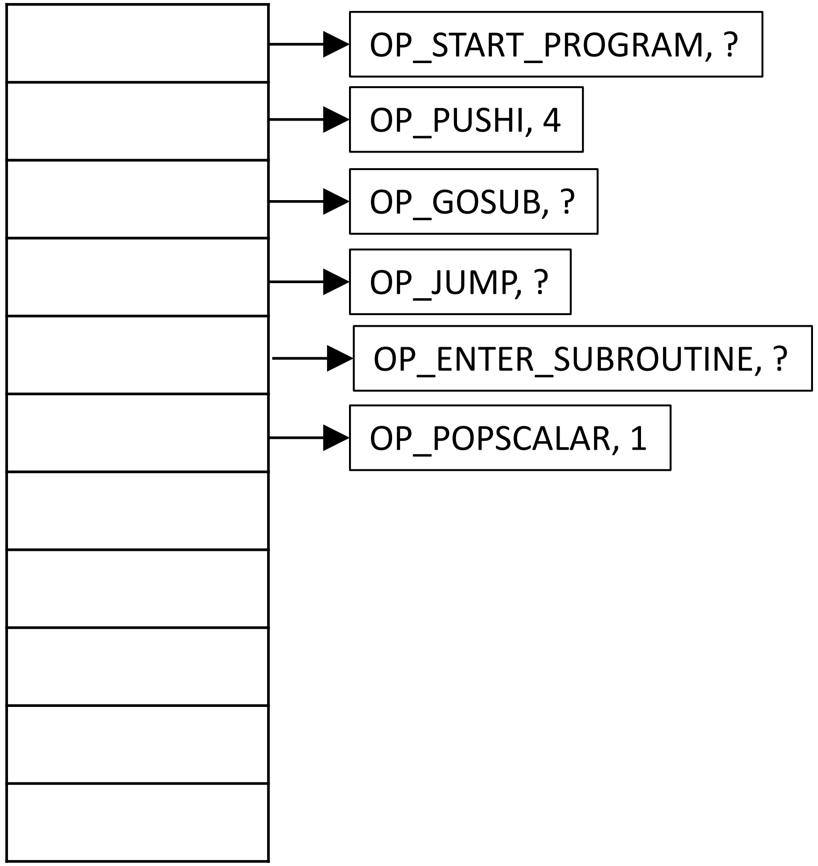
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**A <1, 0>**

String buffer: empty

statement buffer:



- The *popscal A* statement causes a PopScalar object to be created and pointed to by the instruction buffer.
- The operand will be the location of A in the data memory.
- Since variables are always declared before being used, A can be looked up in the symbol table and its location (1) determined immediately.

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**A <1, 0>**

String buffer: empty

statement buffer:

- OP_START_PROGRAM, ?
- OP_PUSHI, 4
- OP_GOSUB, ?
- OP_JUMP, ?
- OP_ENTER_SUBROUTINE, ?
- OP_POPSCALAR, 1
- OP_PUSHSCALAR, 1

- The *pushscal A* statement causes a PushScalar statement object to be created and pointed to by the instruction buffer.
- The operand will be the location of A. Since variables are always declared before being used, A can be looked up in the symbol table and its location (1) determined.

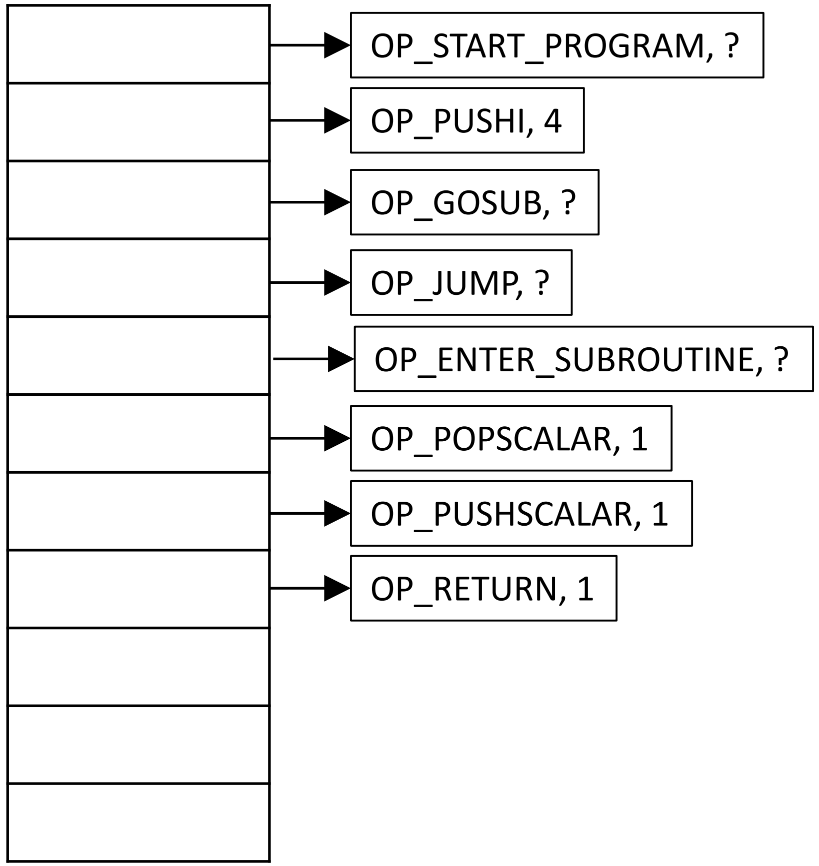
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**

String buffer: empty

statement buffer:



- The *Return* statement causes a Return statement object to be created and pointed to by the instruction buffer.
- The operand of the OP_ENTER_SUBROUTINE statement is the combined length all variables declared in the subroutine and gotten from the symbol table. This allows storage space in the data memory to be allocated for the variables.
- The symbol table for the subroutine is popped or deleted. The operand is not used and is undefined.

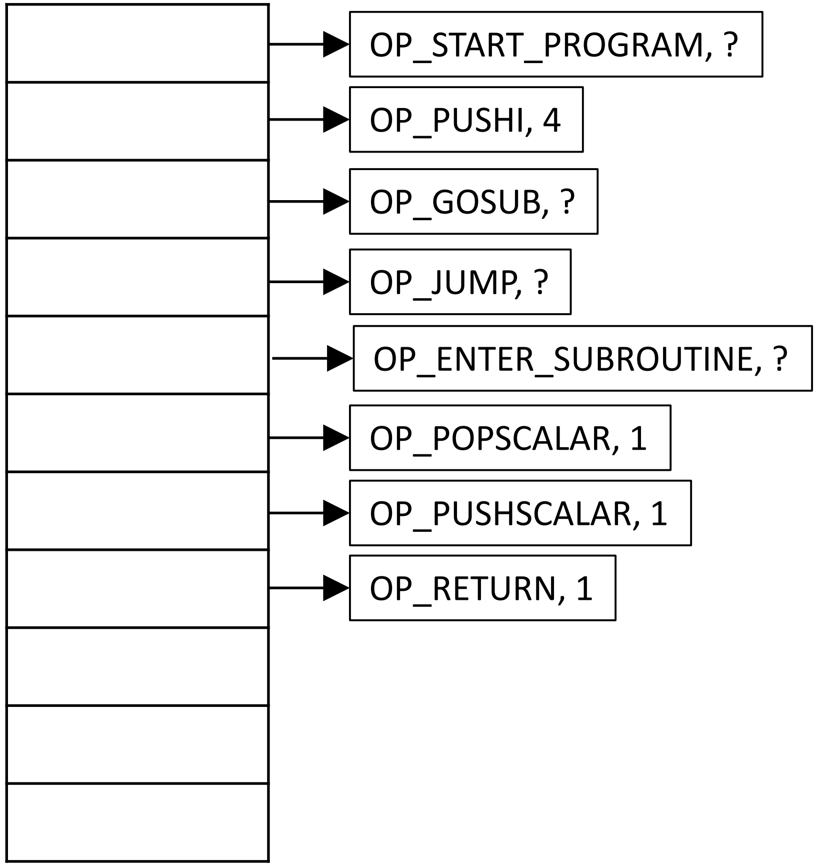
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**L2 <8, 0>**

String buffer: empty

statement buffer:



- The *label L2* statement causes a symbol table entry for the label to be created.
- Its location is the location of the next instruction *in the instruction buffer* following the label, 8 in this case.
- Its length is 0.

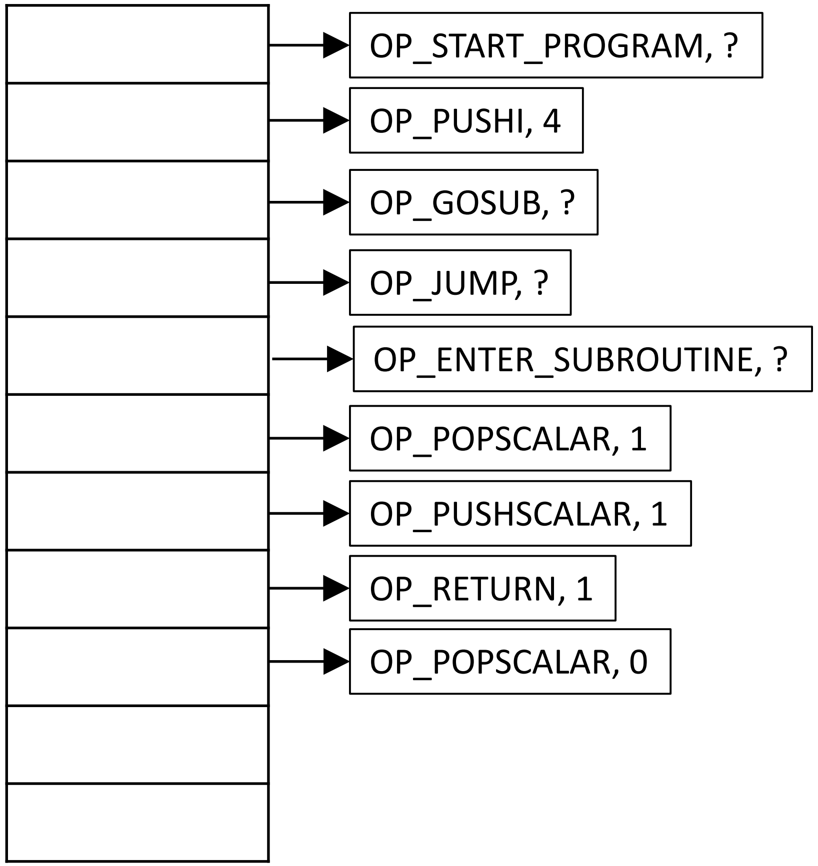
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**L2 <8, 0>**

String buffer: empty

instruction buffer:



- The *popscal A* statement causes a PopScalar statement object to be created and pointed to by the instruction buffer.
- A is looked up in the symbol table, and is now the A declared in the outer scope.
- It’s location is the operand of the popscal instruction

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

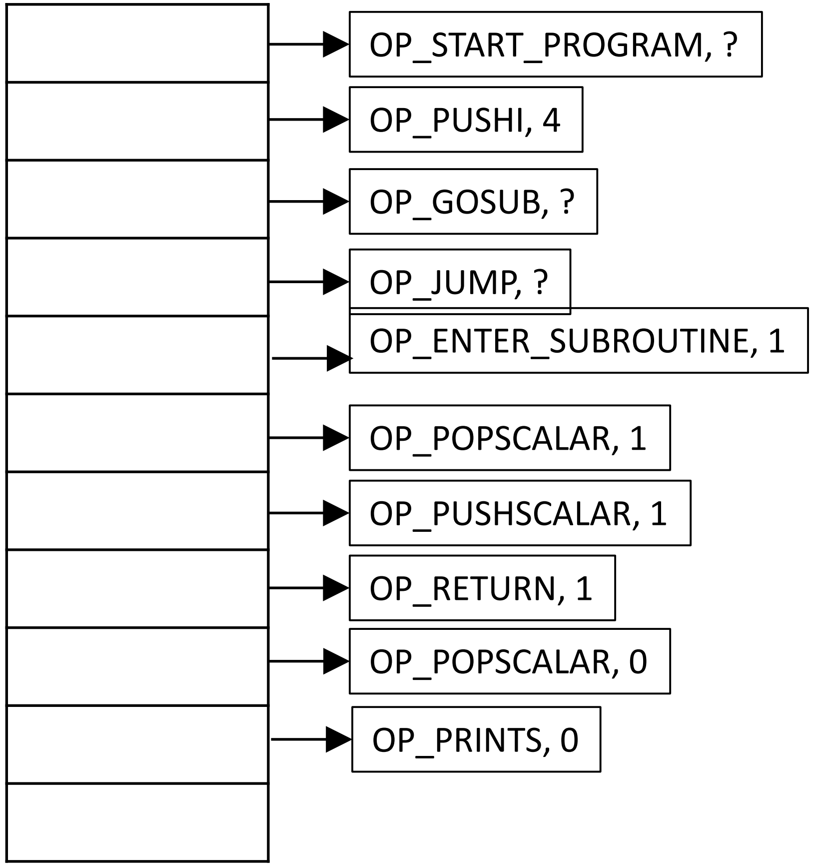
Symbol table:

A, <0,1>	**L1 <4, 0>**	**L2 <8, 0>**

String buffer:

exit_pgm

instruction buffer:



- The *prints exit_pgm* statement causes a Prints statement object to be created and pointed to by the instruction buffer.
- The string "exit_pgm" is added to the string buffer, and its location (0) is the operand of the instruction.

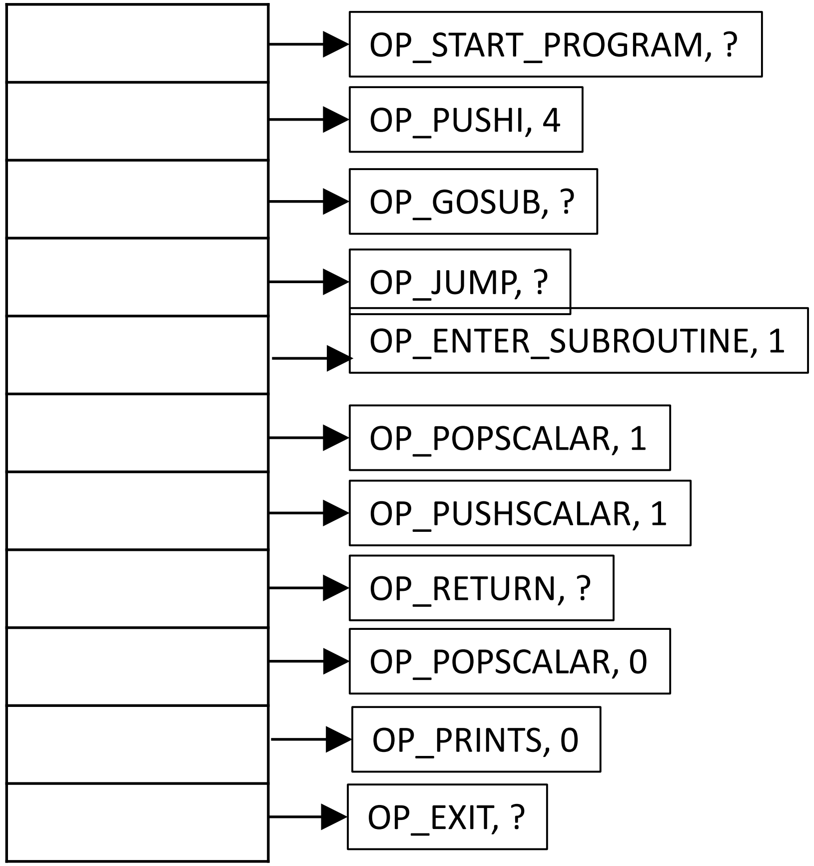
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**L2 <8, 0>**

String buffer: **exit_pgm**

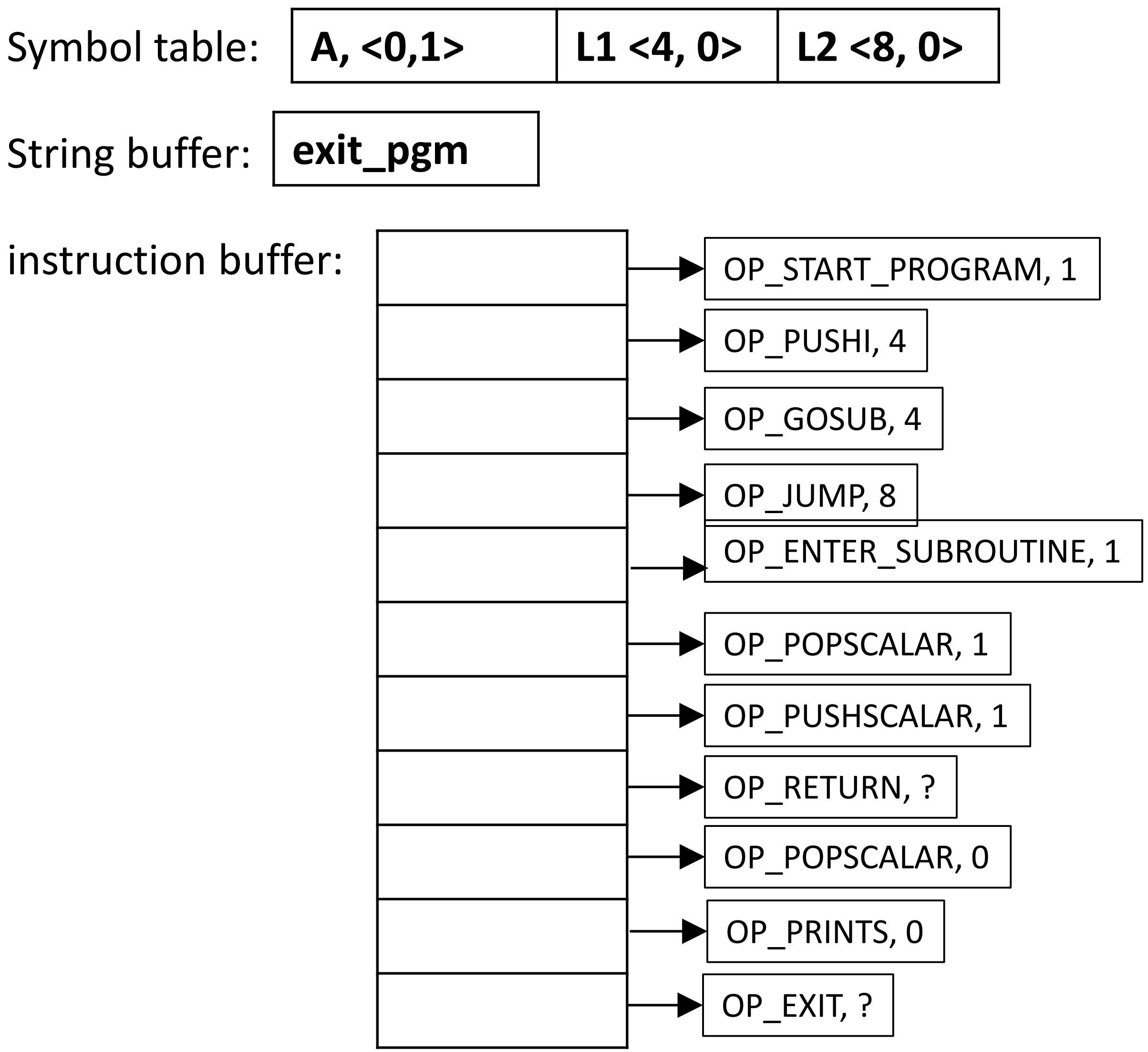
instruction buffer:



- The *exit* statement causes an Exit statement object to be created and pointed to by the instruction buffer.
- Its operand is unused and left undefined.

OP_ENTER_SUBROUTINE, 1

```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```



- The *end* statement signals the end of the program text
- All symbols have been defined.
- The parser now goes through all statements with undefined operands that point to labels, and patches them up.
  - Thus, the OP_JUMP and OP_GOSUB statements are patched.

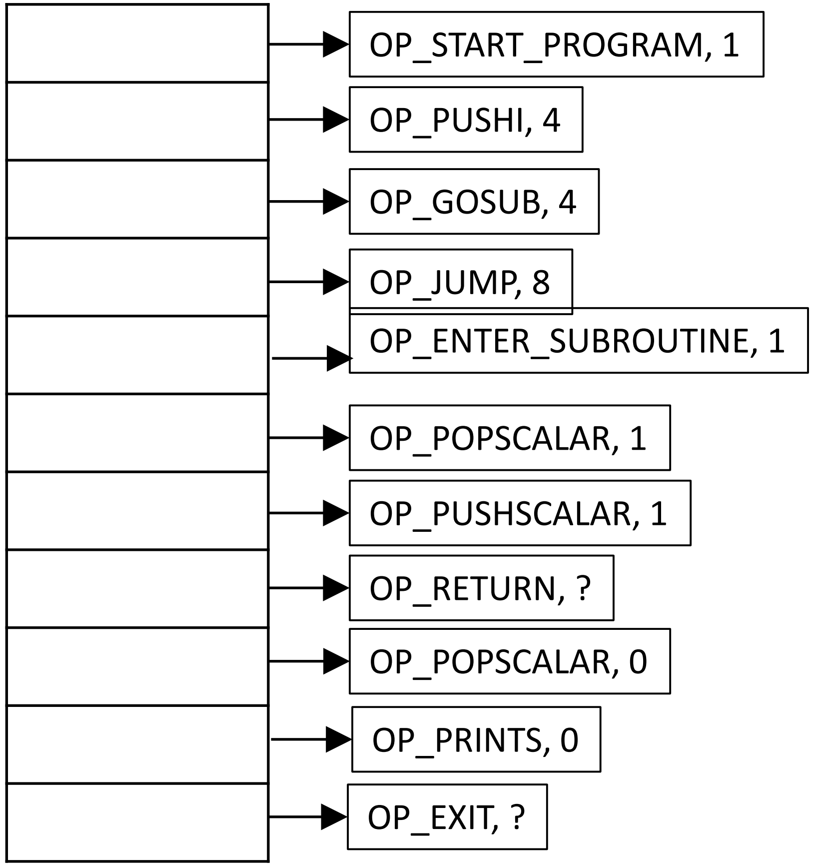
```
start
declscal A
pushi 4
gosub L1
jump L2
gosublabel L1
declscal A
popscal A
pushscal A
return
label L2
popscal A
prints exit_pgm
exit 0
end
```

Symbol table:

A, <0,1>	**L1 <4, 0>**	**L2 <8, 0>**

String buffer: **exit_pgm**

instruction buffer:



- The OP_START_PROGRAM operand is patched with the size of all variables declared in the outer scope to allow the initial stack frame to be set up at runtime.
- The string table and the instructions are dumped to the .out file to be passed to the VM for execution.
- Now, the parser has completed its work.