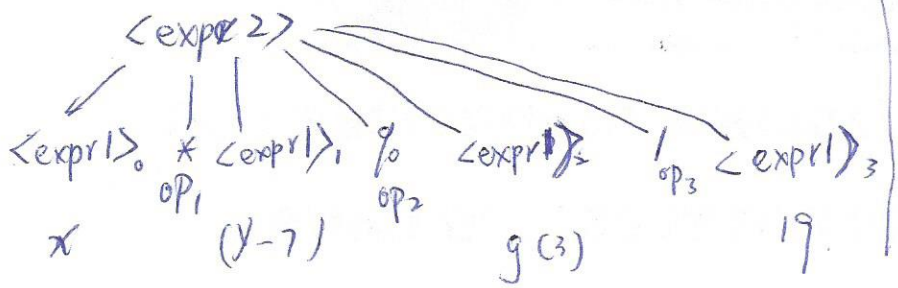
Example of an <expr2>:

x + (y - 7) % g(3) / 19



Assume: variable
x is static with address 3
Variable y is local with stack frame offset +4
g is a method whose code starts at address 29.

In this ~~exp~~ example

<expr2>.code = <expr1>0.code
PUSHSTATADDR 3.
LOADFROMADDR

<expr1>2.code
~~PUSH~~ PUSHSTATADDR 4
LOADFROMADDR.
PUSHNUM 7
SUB

~~<expr~~ OP1.instr
MUL

<expr1>2.code
PUSHNUM 3
PASSPARAM.
CALLSTATMETHOD 29

OP2.instr
MOD

<expr1>3.code.
PUSHNUM 19

OP3.instr
~~PA~~ DIV

new ADDinstr() generates ADD.

~~new PUSHNUMINST~~

new PUSHNUMinstr(23) generates PUSHNUM 23.

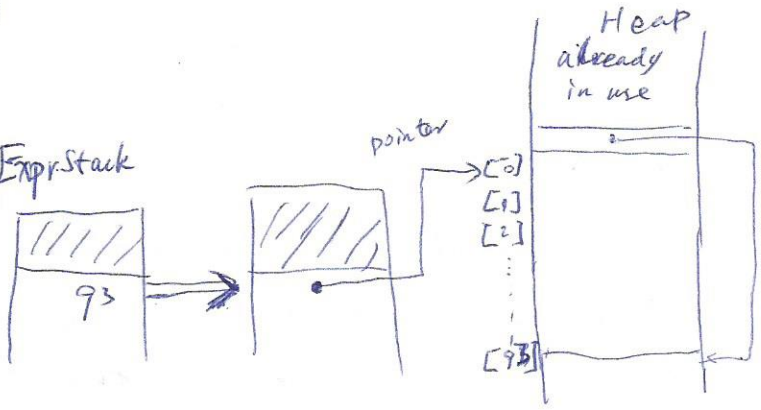
new WRITESTRINGinstr(4, 12) generates 4 12.

@ in ~~<expr2>.whole thing~~

See Parse AND. Translator.java.Txt. <expr2> changes.

CS 316 5/5/2014 ②

new int[93]



Code generated:

<expr3>.code = PUSHNUM 93
HEAPALLOC

while (x < 4) {

/Instruction: getNextCodeAddress()
returns the code memory address of the next instruction to be ~~opp~~ generated

The metod whileStmt() should:

1. Save Instruction.getNextCodeAddress() in an int variable a before calling expr7().
2. Generate the Jumponfalse using something like:

JUMPONFALSEinstr jFinstr = new JUMPONFALSEinstr(Instruction.OPERAND_NOT_YET_KNOWN)

3. After calling statement(), generate

JUMP a

using the variable a of 1.

4. Fixup the JUMPONFALSE using

jFinstr.fixUpOperand(Instruction.getNextCodeAddress())

/AssignmentOr/Invoc.:

x=19; PUSHSTA ADDR 3
PUSHNUM 19
SAVE TO ADDR