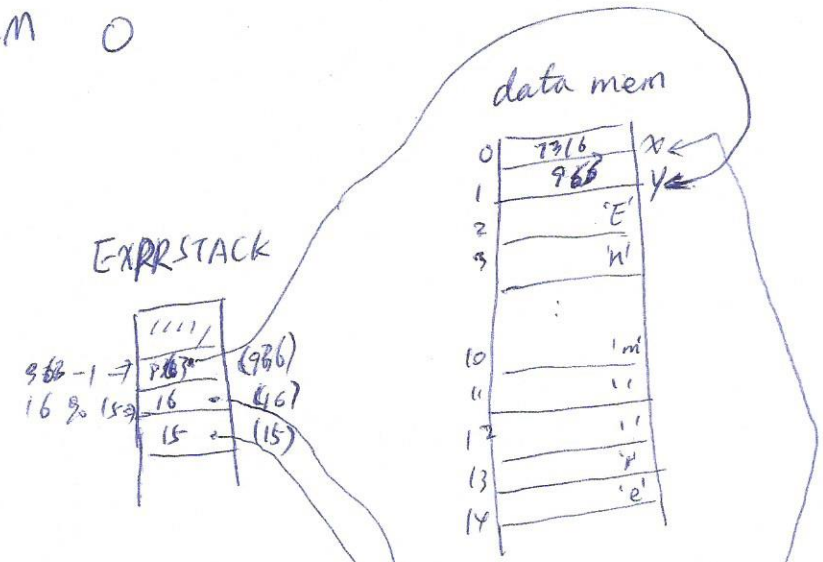
main() is translated to: INITSTKFRM 0

```
INITSTKFRM   0
WRITESTRING  2   12
PUSHSTATADDR 0
READINT
SAVETOADDR
```

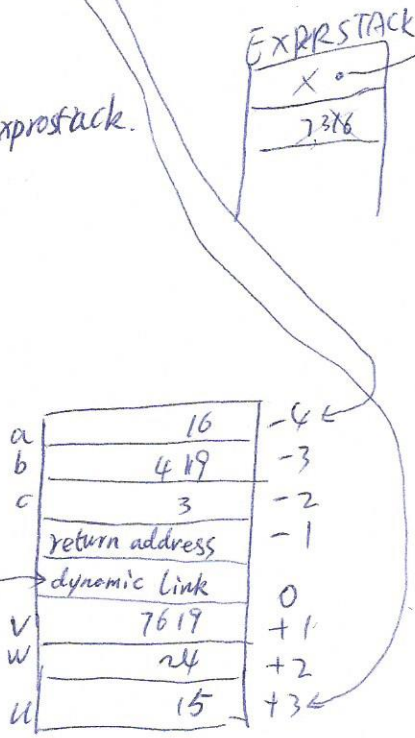


evaluate expression: y - a % u

Translation of return y - a % u; in method f()

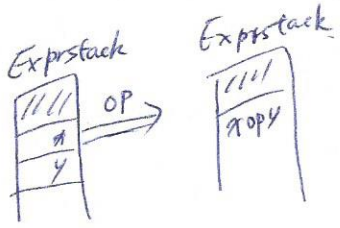
{ instruction that leave value of y - a % u on top of Exprstack.
return 3.

stack frame of each call of f() is as on follows.



To leave value of y - a % u on top of Exprstack.

```
f1: PUSHSTATADDR 1.
    LOADFROMADDR
    PUSHLOCADDR -4
    LOADFROMADDR
    PUSHLOCADDR +3
    LOADFROMADDR
    MOD
f8: SUB
```



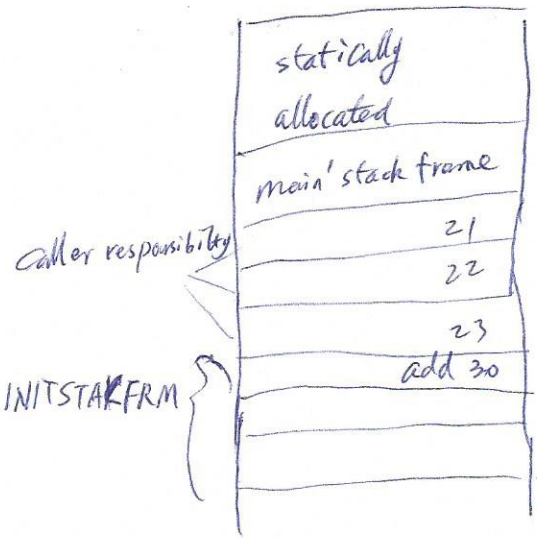
OP Maybe: ADD, SUB, DIV, MUL
MOD,
AND, OR
EQ, NE, LE, LT, GE, GT

Translation of f(21,22,23)

```
PUSHNUM 21
PASSPARAM
PUSHNUM 22
PASSPARAM
PUSHNUM 23
PASSPARAM
CALLSTATMETHOD 34.
```

in main()

Data MEM



Translation of

y = f(21, 22, 23).

```
PUSHSTATADDR 1
LOADFROMADDR
Code to call f(21, 23.23)
ADD
```

In Assignment 1, call a parsing method n() will read an <n> and output a parse tree with root <n> for that instance of <n>

In Assignment 2, that call of n() will also generate <n>.code for that instance of <n>.

Translation of

f(17, y, (x-y));

```
push NUM 17
passPARAM
y { pushSTATADDR 1
    LODASTATADDR
    PASSPARAM
x-y { PUSHSTATADDR 0
      LODASTATADDR
      PUSHSTATADDR 1
      LODASTATADDR
      SUB
      PASSPARAM.
```

CallSTATMETHOD 34

DISCARDVALUE.

Notation:

For any node <n> in the TinyJ same file parse tree, let

<n> code

mean the VM instruction generated by the compiler for <n>.

Ex: if <n> is the <statement> node for

f(17, y, x-y)

in the above program, then <n>.code = instructions 8-20 in the generated code