i116: Basic of Programming

13. Programming language processor: virtual machine

Kazuhiro Ogata, Canh Minh Do

Roadmap

- Virtual machine for Minila

Virtual machine for Minila

- It does not suffice to handle a list of commands from the top to the bottom.
- It is necessary to have a command that deals with a condition so that **if** and **while** statements can be handled.
- It is also necessary to go (or jump) forward or backward to a position in such a list to this end.
- We may want to have a command that makes a program halt.

Virtual machine for Minila

- We add the following three new commands:
  - jmp(*n*)
  - cjmp(*n*)
  - quit

  where *n* is an integer.

  jmp stands for jump.

  cjmp stands for conditional jump.

Virtual machine for Minila

```
from enum import *
class CName(Enum):
    ....
    OR = auto()
    JMP = auto()
    CJMP = auto()
    QUIT = auto()
    NSC = auto()
```

```
def __str__(self):
    ...
    elif self == CName.OR:
        return 'or'
    elif self == CName.JMP:
        return 'jmp'
    elif self == CName.CJMP:
        return 'cjmp'
    elif self == CName.QUIT:
        return 'quit'
    ...
```

Adding three command names: JMP , CJMP and QUIT

Virtual machine for Minila

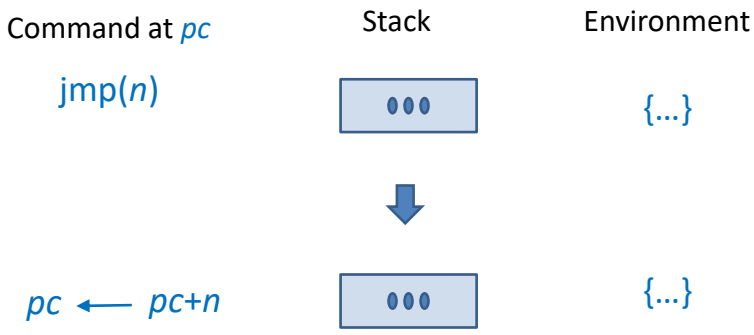
```
class Command(object):
    ...
    def __init__(self, cn, x):
        ...
        elif cn == CName.JMP:
            self.num = x
        elif cn == CName.CJMP:
            self.num = x
    def __str__(self):
        ...
        elif self.cname == CName.JMP:
            return str(self.cname) + '(' + str(self.num) + ')'
        elif self.cname == CName.CJMP:
            return str(self.cname) + '(' + str(self.num) + ')'
        else:
            return str(self.cname)
```

Adding three commands:
jmp(*n*), cjmp(*n*) and quit

Virtual machine for Minila

- Instead of handling commands from the top to the bottom in a list of command, we will use a program counter (*pc*) to decide what command will be handled next.
- *pc* is initially 0, referring to the top command in such a list.
- If quit is the command at the position referred to by *pc*, the VM quits and returns the environment.

Virtual machine for Minila

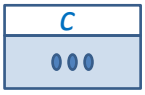


Virtual machine for Minila

Command at *pc*

cjmp(*n*)

Stack



Environment

{...}

If *c* = 0

pc ← *pc*+1

Otherwise

pc ← *pc*+*n*

{...}

If Stack is empty, an exception called VMError is raised.

Virtual machine for Minila

Command at *pc*

quit

Stack

Environment

{...}

{...} is returned as the result

Virtual machine for Minila

Command at *pc*	Stack	Environment
add	*b* *a* …	{…}
pc ⟵ *pc*+1	*a+b* …	{…}

Virtual machine for Minila

Command at *pc*	Stack	Environment
store('x')	*val* …	{…, 'x':*val'*, …} or {…}
pc ⟵ *pc*+1	…	{…, 'x':*val*, …}

Virtual machine for Minila

```
class VM(object):
  ...
  def __init__(self):
    ...
  ....
  def str(self):
    return 'pc: ' + str(self.pc) + ', stack: ' ...
  def run(self):
    while True:
      if self.pc < 0 or self.pc >= len(self.clist):
        raise VMError('pc: ' + str(self.pc) + ' is out of scope of clist: ' + l2s(self.clist))
      com = self.clist[self.pc]
      if com.cname == CName.PUSH:
        self.stk = self.stk.push(com.num)
        self.pc = self.pc + 1
```


Virtual machine for Minila

```
      elif com.cname == CName.LOAD:
        ...
        self.pc = self.pc + 1
      elif com.cname == CName.STORE:
        ...
        self.pc = self.pc + 1
      elif com.cname == CName.MONE:
        ...
        self.pc = self.pc + 1
      elif com.cname == CName.MUL:
        ...
        self.pc = self.pc + 1
      elif com.cname == CName.QUO:
        ...
        self.pc = self.pc + 1
```

Virtual machine for Minila

```
elif com.cname == CName.REM:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.ADD:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.SUB:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.EQ:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.NEQ:
    ...
    self.pc = self.pc + 1
```


Virtual machine for Minila

```
elif com.cname == CName.LT:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.GT:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.AND:
    ...
    self.pc = self.pc + 1
elif com.cname == CName.OR:
    ...
    self.pc = self.pc + 1
```

Virtual machine for Minila

```
elif com.cname == CName.JMP:
    self.pc = self.pc + com.num
elif com.cname == CName.CJMP:
    if self.stk.isEmpty():
        raise VMError('stk is empty for cjmp')
    x = self.stk.top()
    self.stk = self.stk.pop()
    if x == 0:
        self.pc = self.pc + 1
    else:
        self.pc = self.pc + com.num
elif com.cname == CName.QUIT:
    return self.env
else:
    raise VMError("An invalid command was met!")
```


Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{ }

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

1

{ }

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

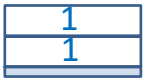
{'x':1,'y':1}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]



{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

Virtual machine for Minila

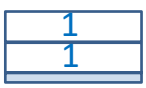
[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]



{'x':1,'y':1}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':1}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

Virtual machine for Minila

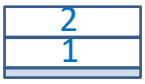
[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]



{'x':1,'y':2}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':2}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':1,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':2}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':2,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':3}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':3}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':3}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':6,'y':4}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':4}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':4}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

Virtual machine for Minila

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

[push(1), store(x), push(1), store(y), load(y), push(5), lt, cjmp(2), jmp(10),
load(x), load(y), mul, store(x), load(y), push(1), add, store(y), jmp(-13), quit]

{'x':24,'y':5}

{'x':24,'y':5} is returned as the result.

Virtual machine for Minila

```
from vm import *

push1 = Command(CName.PUSH,1)
storeX = Command(CName.STORE,'x')
storeY = Command(CName.STORE,'y')
push5 = Command(CName.PUSH,5)
lt = Command(CName.LT,None)
cjmp2 = Command(CName.CJMP,2)
jmp10 = Command(CName.JMP,10)
loadX = Command(CName.LOAD,'x')
loadY = Command(CName.LOAD,'y')
mul = Command(CName.MUL,None)
add = Command(CName.ADD,None)
jmpM13 = Command(CName.JMP,-13)
quit = Command(CName.QUIT,None)
```


Virtual machine for Minila

```
cl = [push1, storeX, push1, storeY, loadY, push5,
lt, cjmp2, jmp10, loadX, loadY, mul, storeX, loadY,
push1, add, storeY, jmpM13, quit]
print(l2s(cl))
vm = VM(cl)
print(vm.run())
```

Virtual machine for Minila

```
from vm import *

push119 = Command(CName.PUSH,119)
storeX = Command(CName.STORE,'x')
push2 = Command(CName.PUSH,2)
storeY = Command(CName.STORE,'y')
push1 = Command(CName.PUSH,1)
storeR = Command(CName.STORE,'r')
storeF = Command(CName.STORE,'f')
loadF = Command(CName.LOAD,'f')
cjmp2 = Command(CName.CJMP,2)
jmp26 = Command(CName.JMP,26)
loadX = Command(CName.LOAD,'x')
loadY = Command(CName.LOAD,'y')
rem = Command(CName.REM,None)
```


Virtual machine for Minila

```
push0 = Command(CName.PUSH,0)
eq = Command(CName.EQ,None)
jmp6 = Command(CName.JMP,6)
jmp5 = Command(CName.JMP,5)
add = Command(CName.ADD,None)
jmp4 = Command(CName.JMP,4)
jmp1 = Command(CName.JMP,1)
jmpM27 = Command(CName.JMP,-27)
quit = Command(CName.QUIT,None)
```

Virtual machine for Minila

```
cl = [push119, storeX, push2, storeY, push1, storeR,
push1, storeF, loadF, cjmp2, jmp26, loadX, loadY,
rem, push0, eq, cjmp2, jmp6, push0, storeF, push0,
storeR, jmp5, loadY, push1, add, storeY, loadX, loadY,
eq, cjmp2, jmp4, push0, storeF, jmp1, jmpM27, quit]
print(l2s(cl))
vm = VM(cl)
print(vm.run())
```


Virtual machine for Minila

```
from vm import *

push20000000000000000 = Command(CName.PUSH,20000000000000000)
storeV0 = Command(CName.STORE,'v0')
push0 = Command(CName.PUSH,0)
storeV1 = Command(CName.STORE,'v1')
loadV0 = Command(CName.LOAD,'v0')
storeV2 = Command(CName.STORE,'v2')
loadV1 = Command(CName.LOAD,'v1')
loadV2 = Command(CName.LOAD,'v2')
neq = Command(CName.NEQ,None)
cjmp2 = Command(CName.CJMP,2)
jmp44 = Command(CName.JMP,44)
sub = Command(CName.SUB,None)
```

Virtual machine for Minila

```
push2 = Command(CName.PUSH,2)
rem = Command(CName.REM,None)
eq = Command(CName.EQ,None)
jmp10 = Command(CName.JMP,10)
quo = Command(CName.QUO,None)
add = Command(CName.ADD,None)
storeV3 = Command(CName.STORE,'v3')
jmp11 = Command(CName.JMP,11)
loadV3 = Command(CName.LOAD,'v3')
push1 = Command(CName.PUSH,1)
mul = Command(CName.MUL,None)
gt = Command(CName.GT,None)
jmp6 = Command(CName.JMP,6)
jmp3 = Command(CName.JMP,3)
jmpM47 = Command(CName.JMP,-47)
quit = Command(CName.QUIT,None)
```


Virtual machine for Minila

```
cl = [push20000000000000000, storeV0, push0,
storeV1, loadV0, storeV2, loadV1, loadV2, neq,
cjmp2, jmp44, loadV2, loadV1, sub, push2, rem,
push0, eq, cjmp2, jmp10, loadV1, loadV2, loadV1,
sub, push2, quo, add, storeV3, jmp11, loadV1,
loadV2, loadV1, sub, push2, quo, add, push1, add,
storeV3, loadV3, loadV3, mul, loadV0, gt, cjmp2,
jmp6, loadV3, push1, sub, storeV2, jmp3, loadV3,
storeV1, jmpM47, quit]
print(l2s(cl))
vm = VM(cl)
print(vm.run())
```