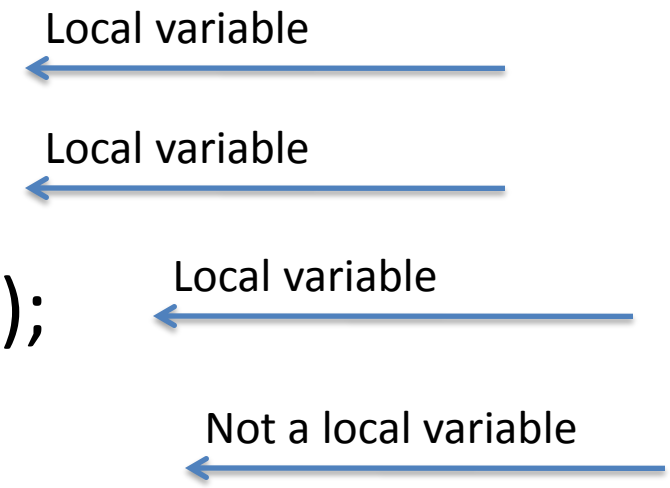
The C0VM

Elias Szabo-Wexler

Example Code

```
int main() {
  int x = 5;
  int y = 2;
  int z = exp(x, y);
  return z;
}
```

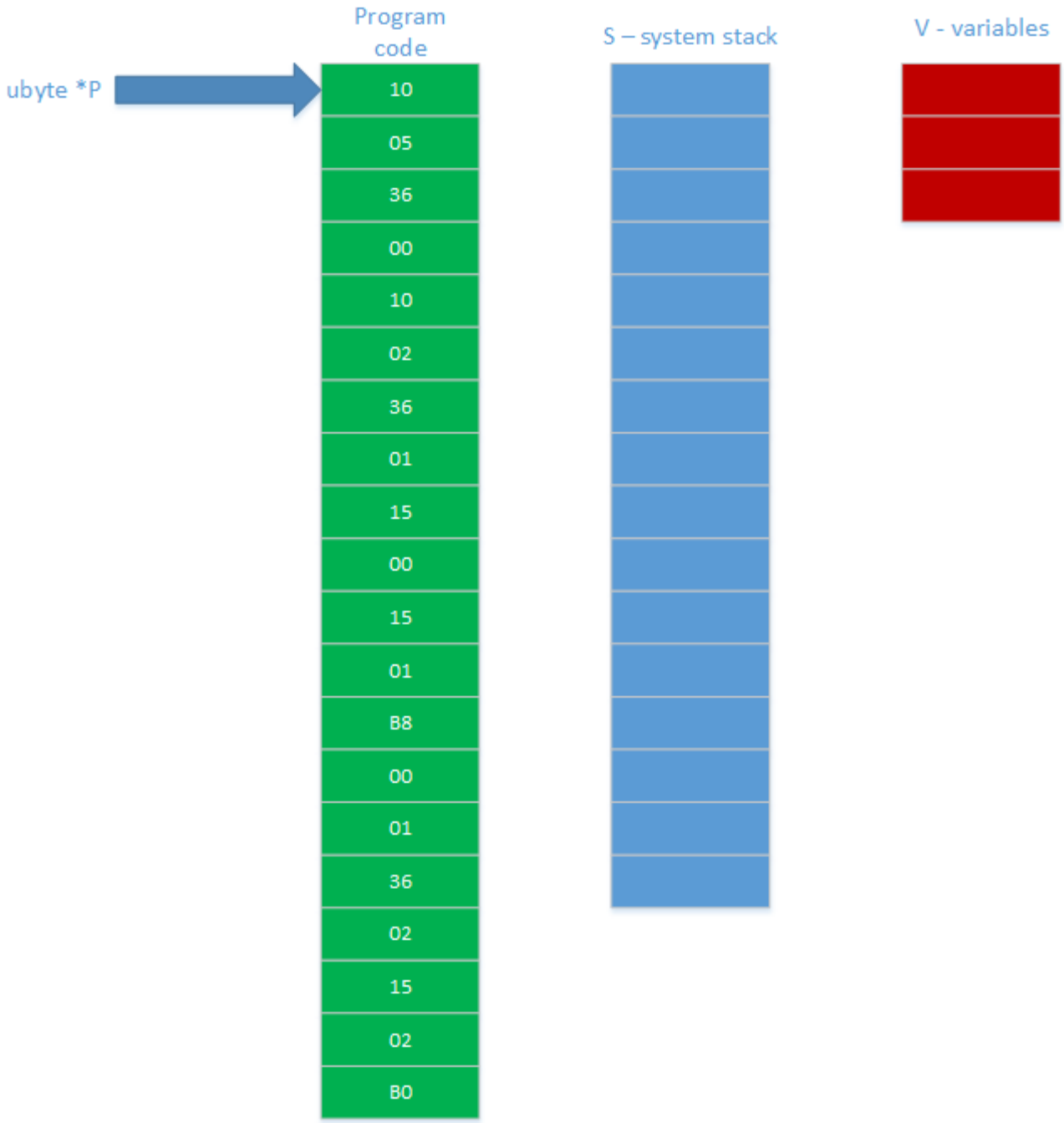


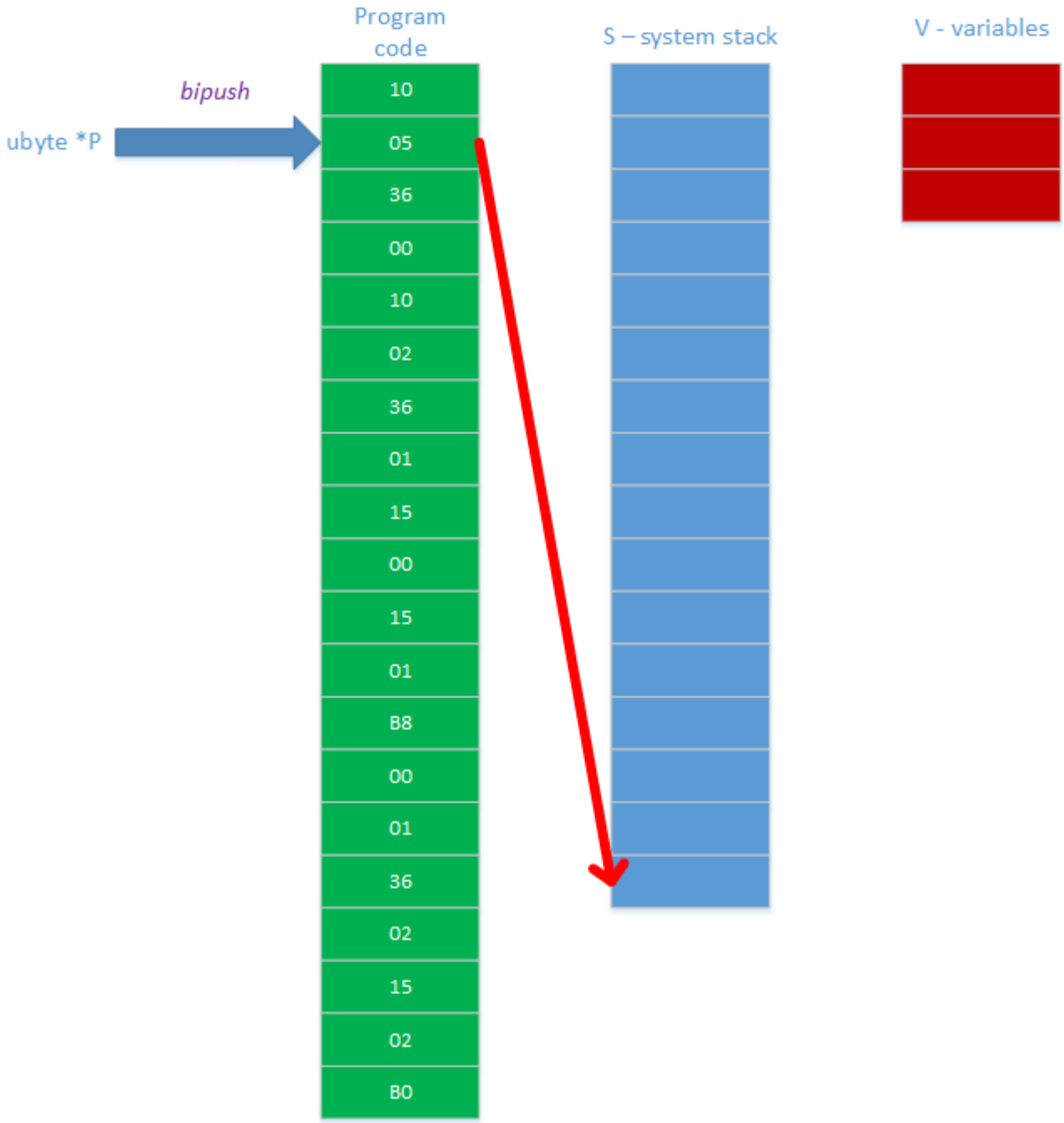
Bytecode

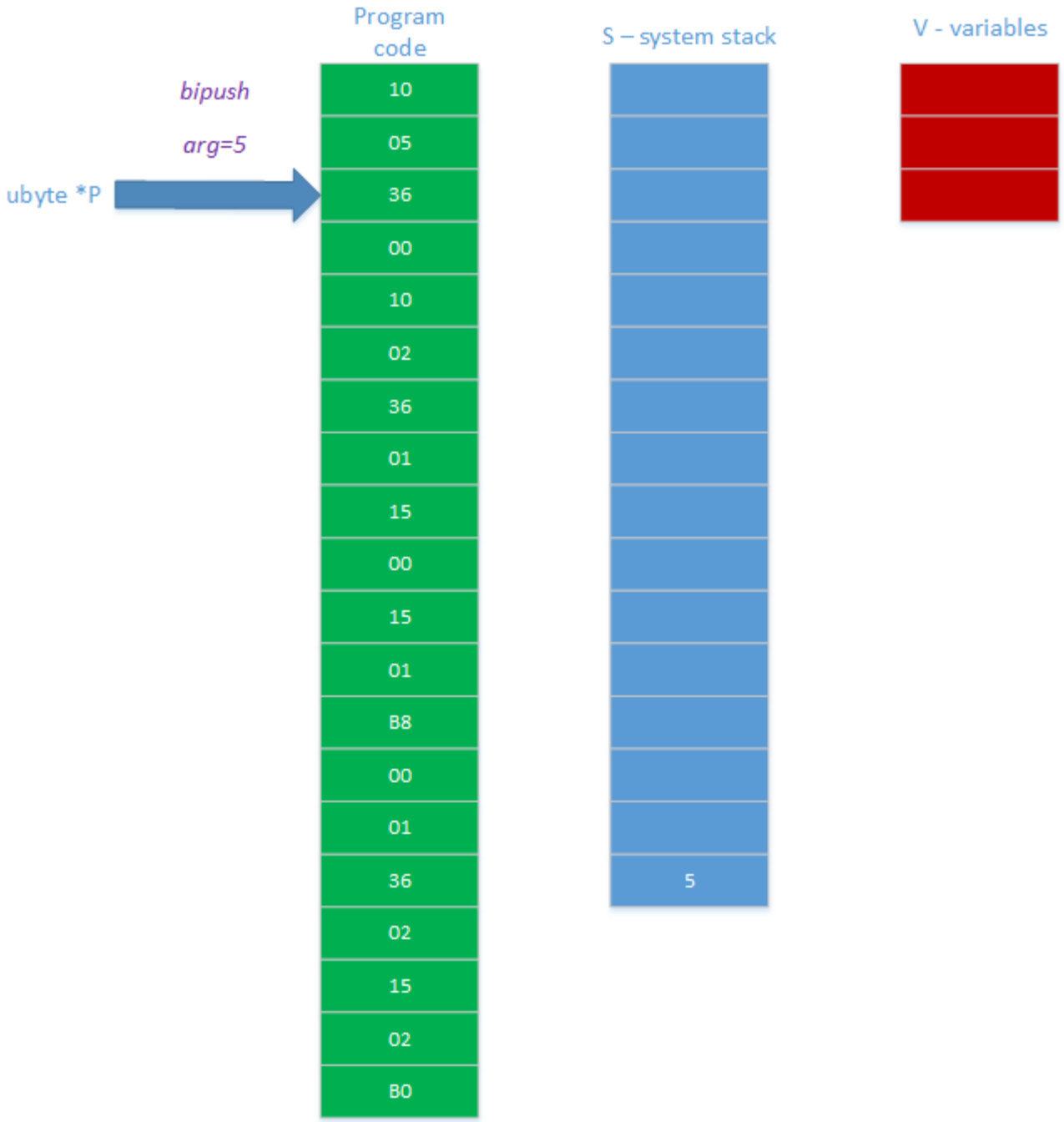
```
#<main>
00 00              # number of arguments = 0
00 03              # number of local variables = 3
00 14              # code length = 20 bytes
10 05     # bipush 5         # 5
36 00     # vstore 0         # x = 5;
10 02     # bipush 2         # 2
36 01     # vstore 1         # y = 2;
15 00     # vload 0          # x
15 01     # vload 1          # y
B8 00 01  # invokestatic 1   # exp(x, y)
36 02     # vstore 2         # z = exp(x, y);
15 02     # vload 2          # z
B0        # return           #
```

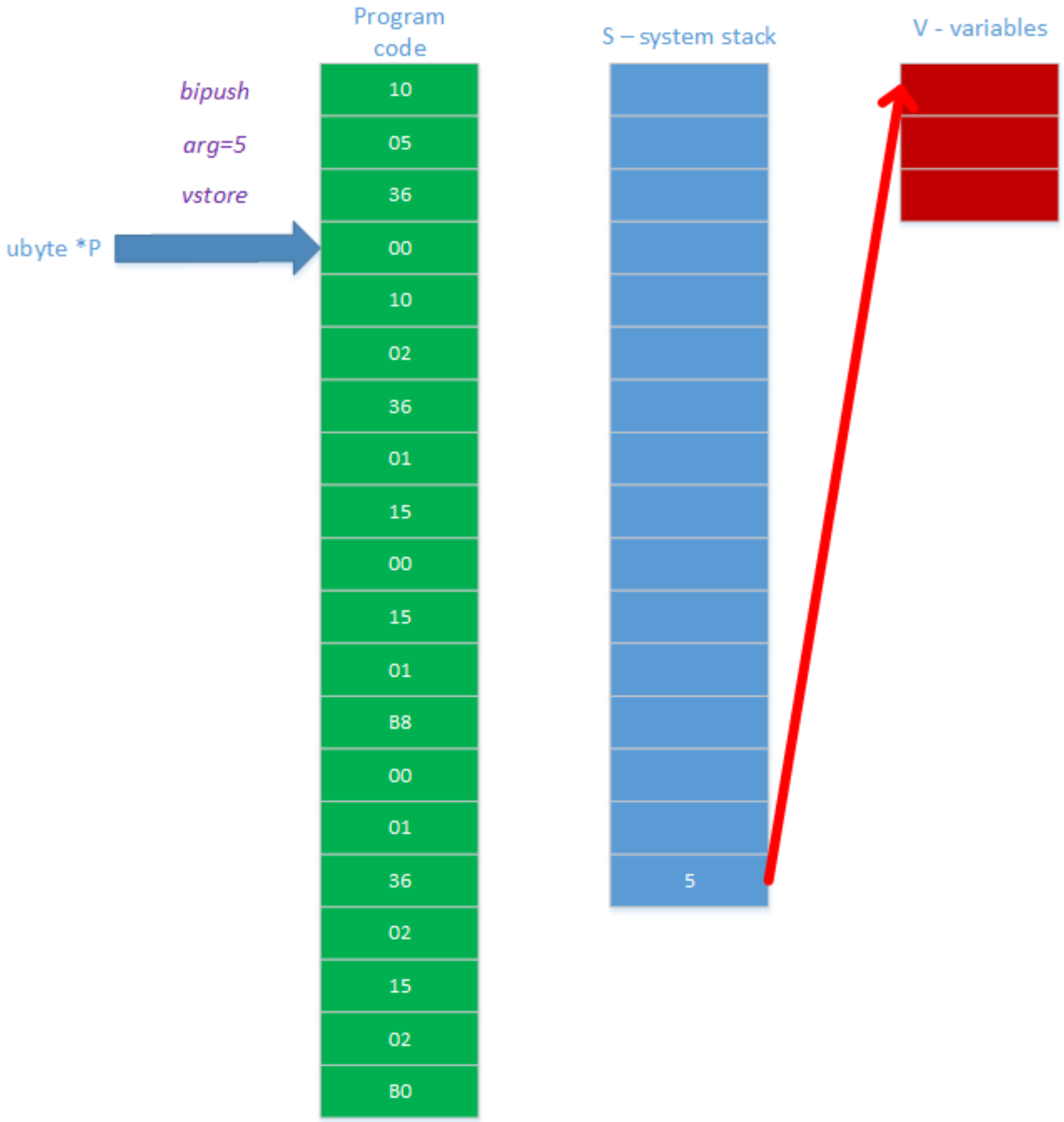
Virtualization

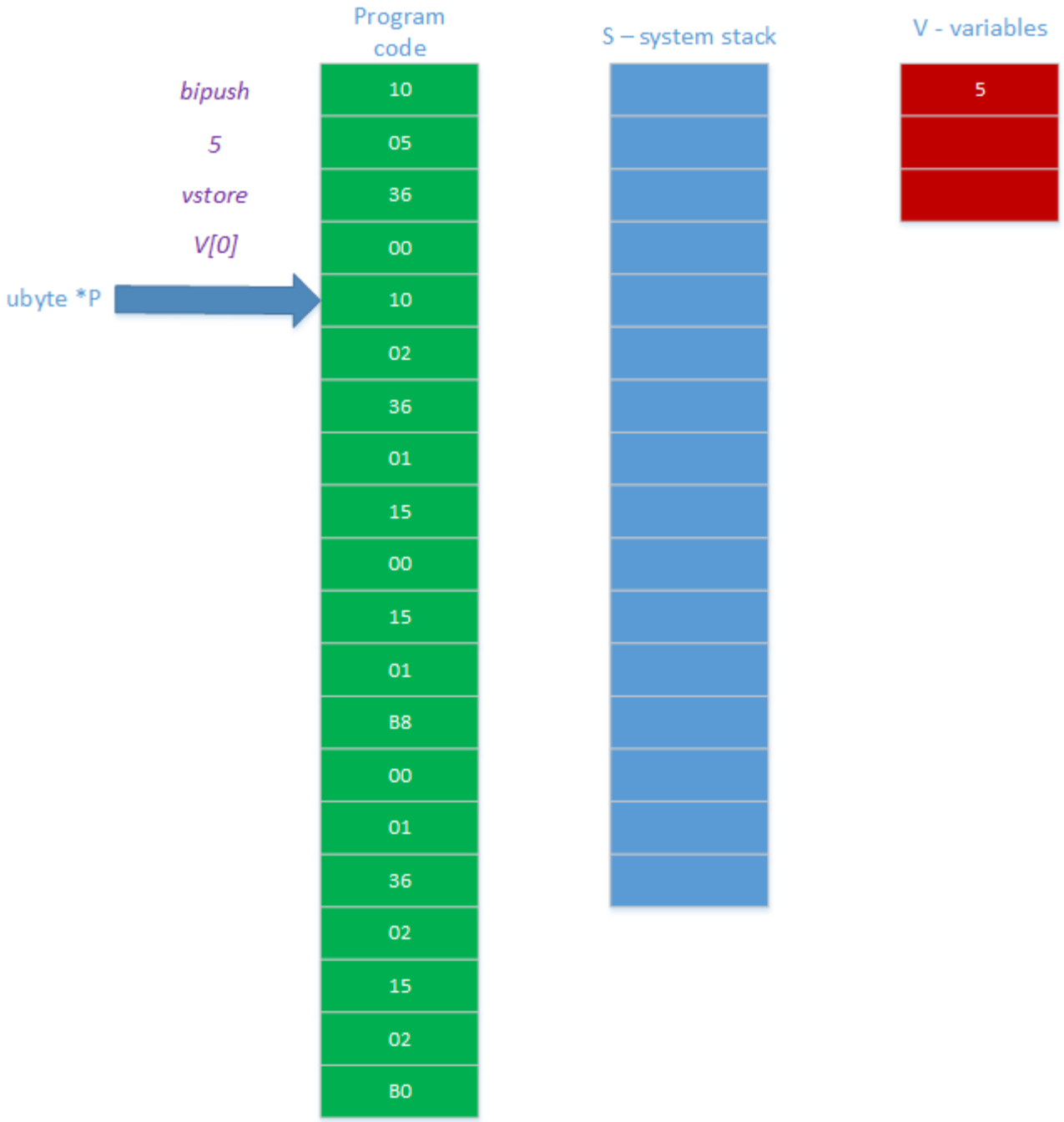
- We'll store the *program* in a byte array
- We'll track execution with a *program counter*
  - ubyte* P
- We'll store *operands* on the system stack
  - stack S = stack_new()
- We'll store *local variables* in an array
  - c0_value* V = xcalloc(num_vars, sizeof(c0_value))
- **All operations use the system stack to communicate!**

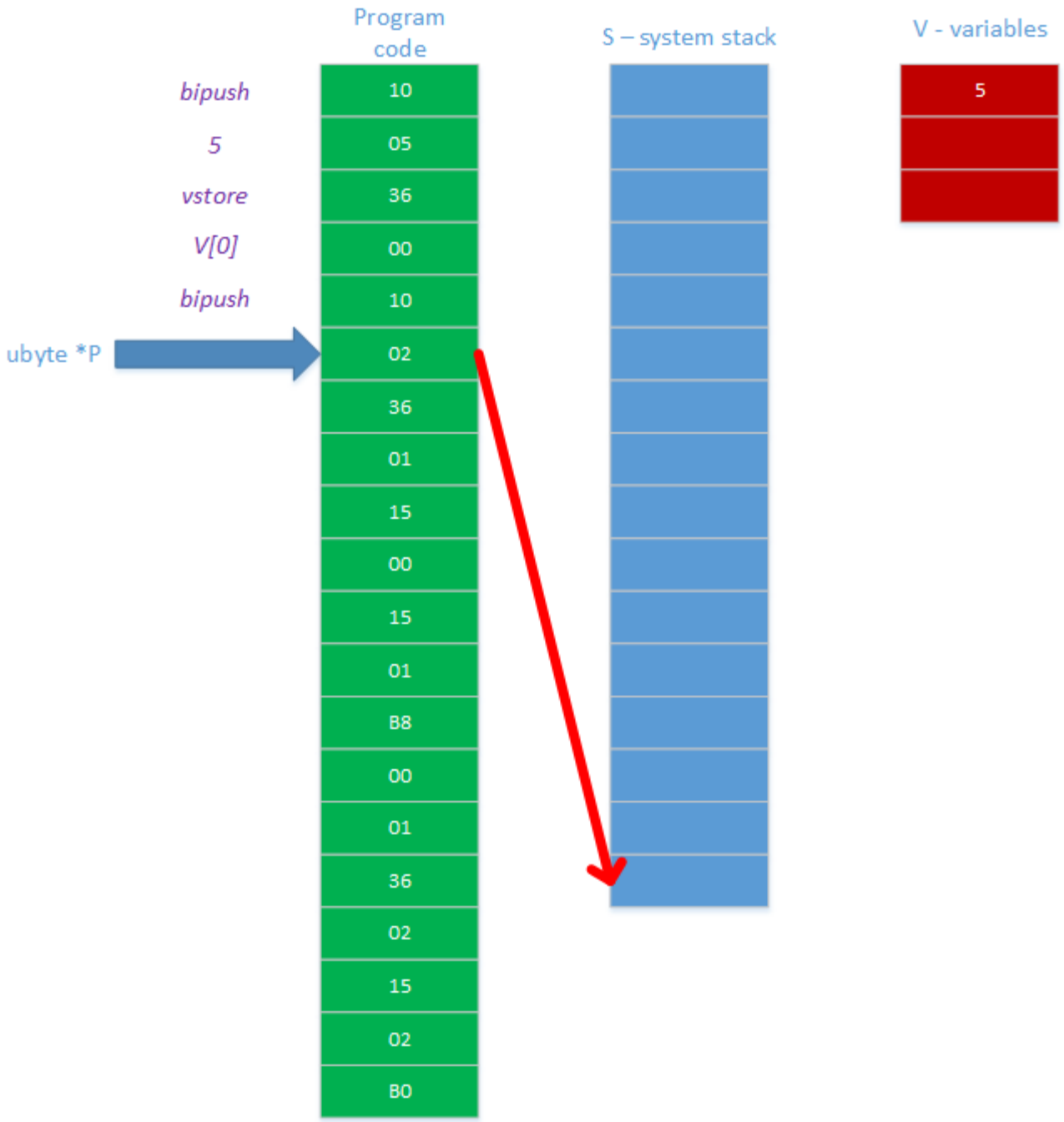

Program code
S – system stack
V - variables
ubyte *P
10
05
36
00
10
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0

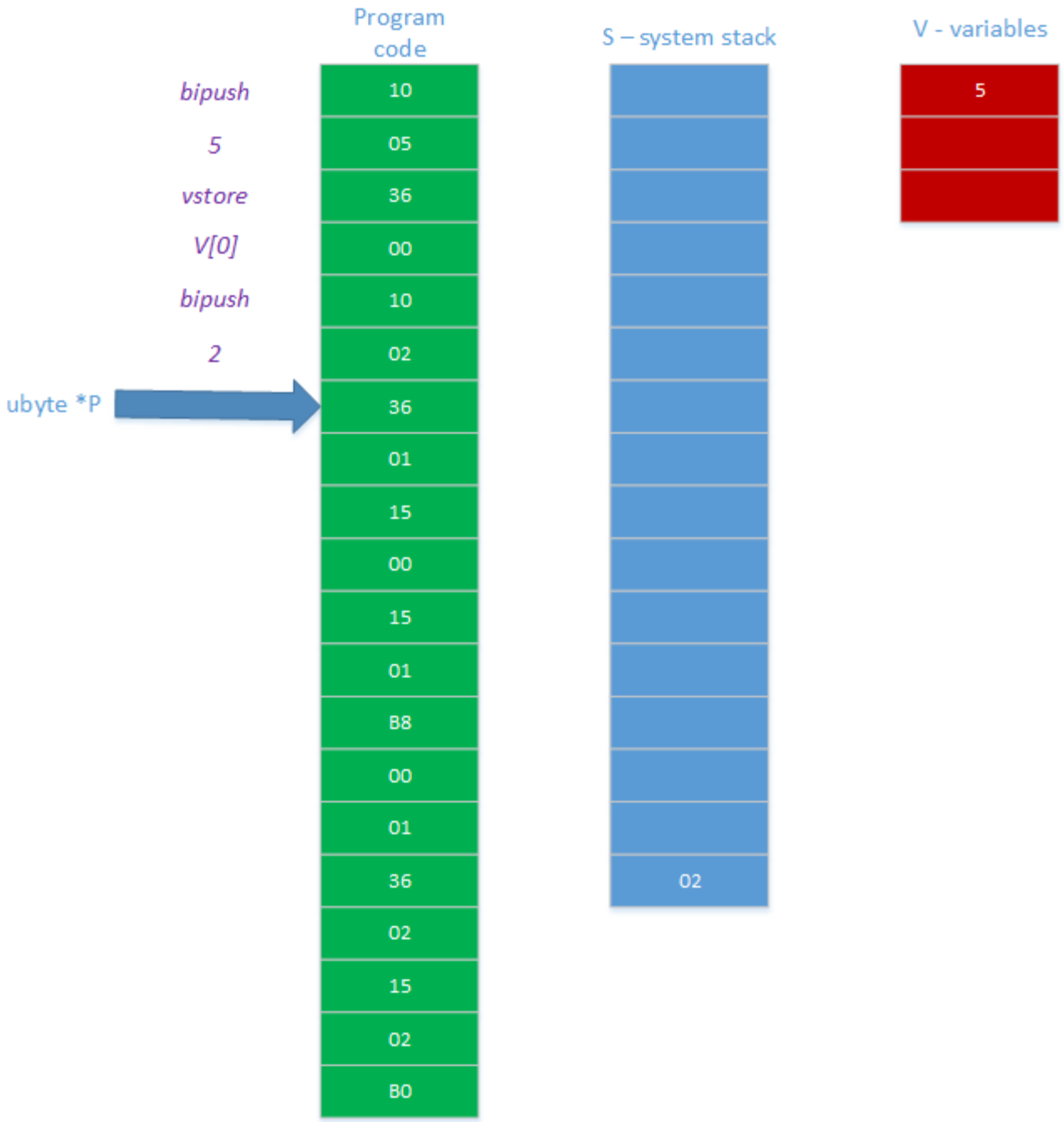

Program code
S – system stack
V - variables
bipush
ubyte *P
10
05
36
00
10
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0

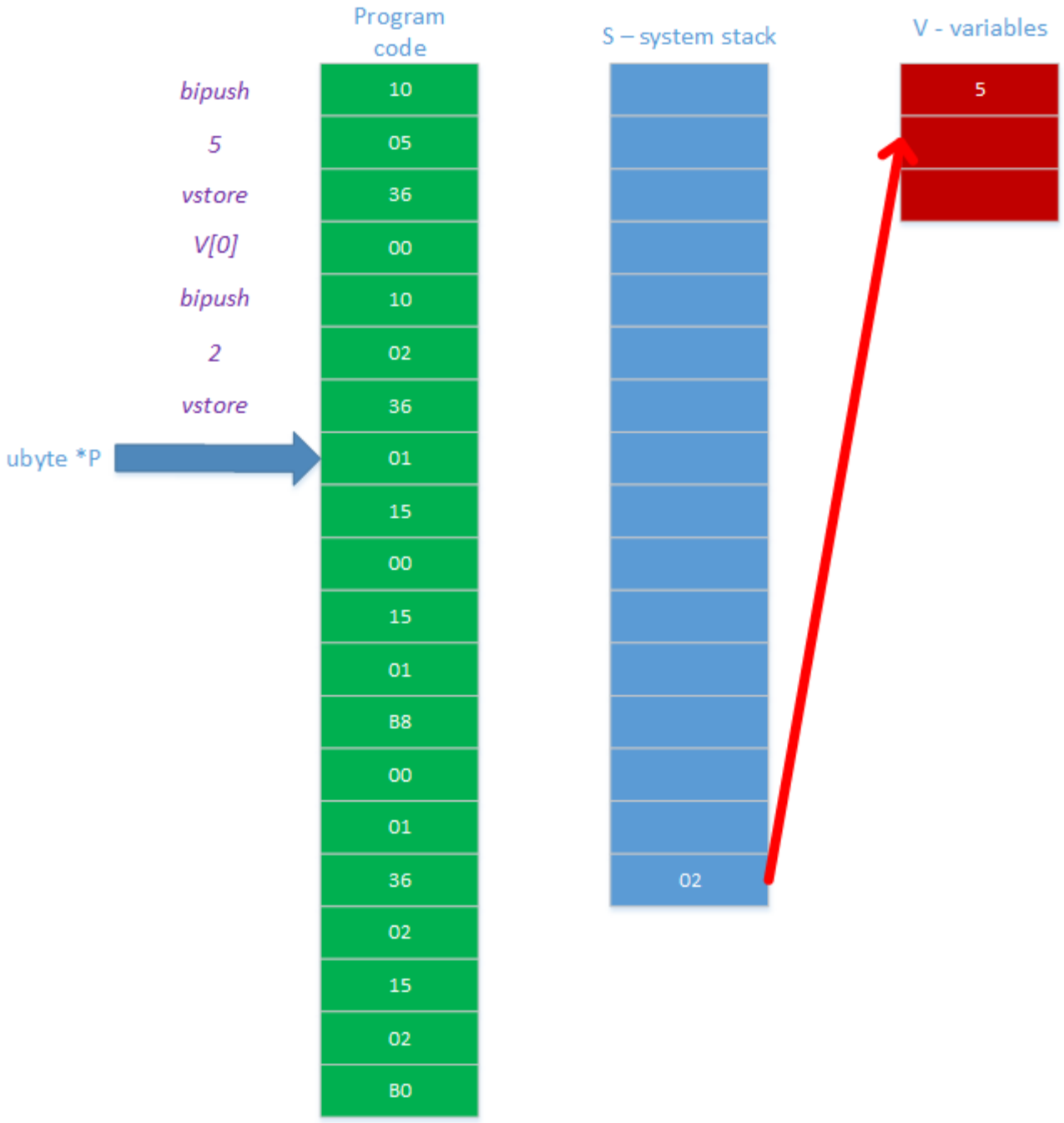

Program code
S – system stack
V - variables
bipush
arg=5
ubyte *P
10
05
36
00
10
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5

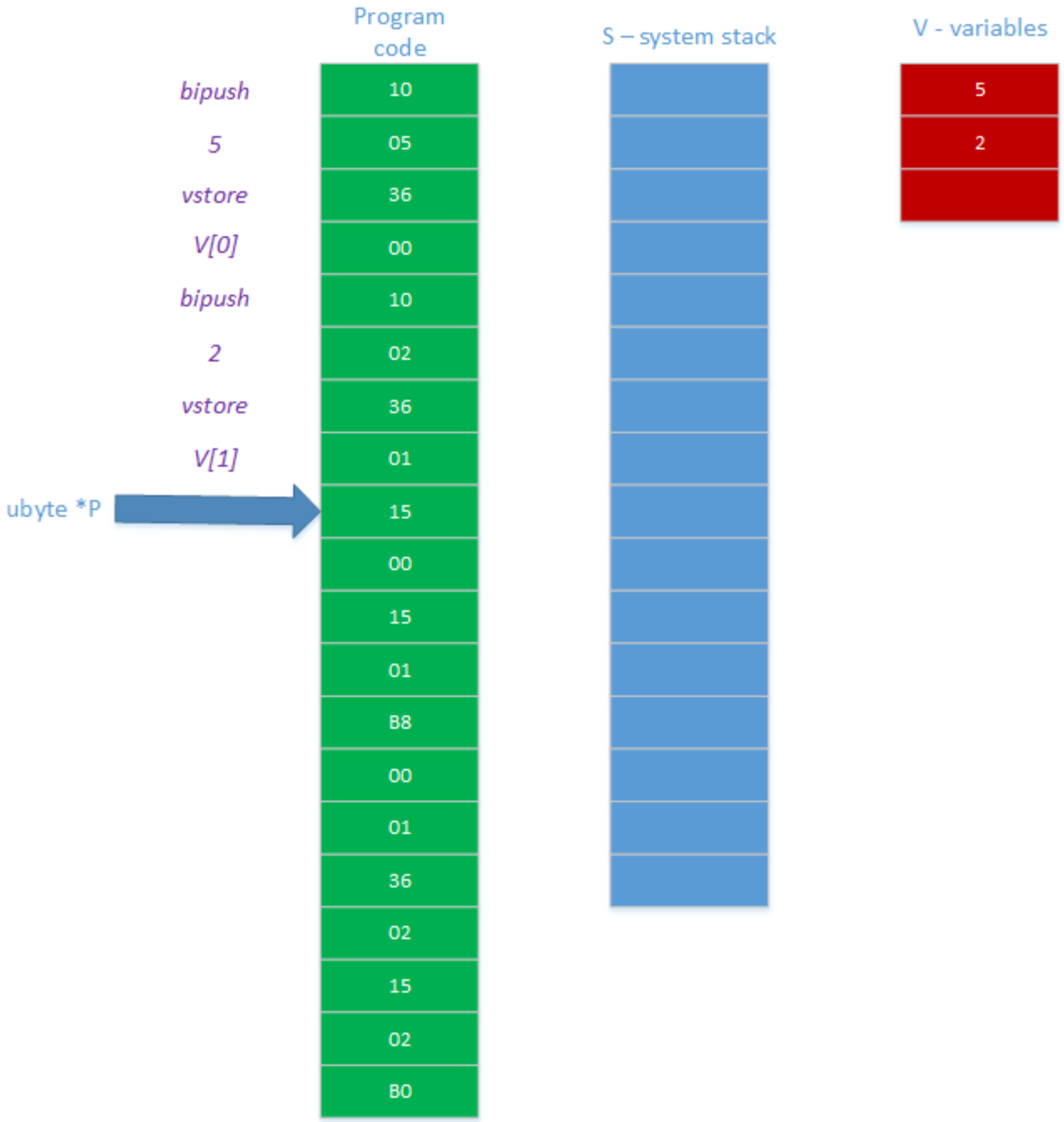

Program code
S – system stack
V - variables
bipush
arg=5
vstore
ubyte *P
10
05
36
00
10
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5

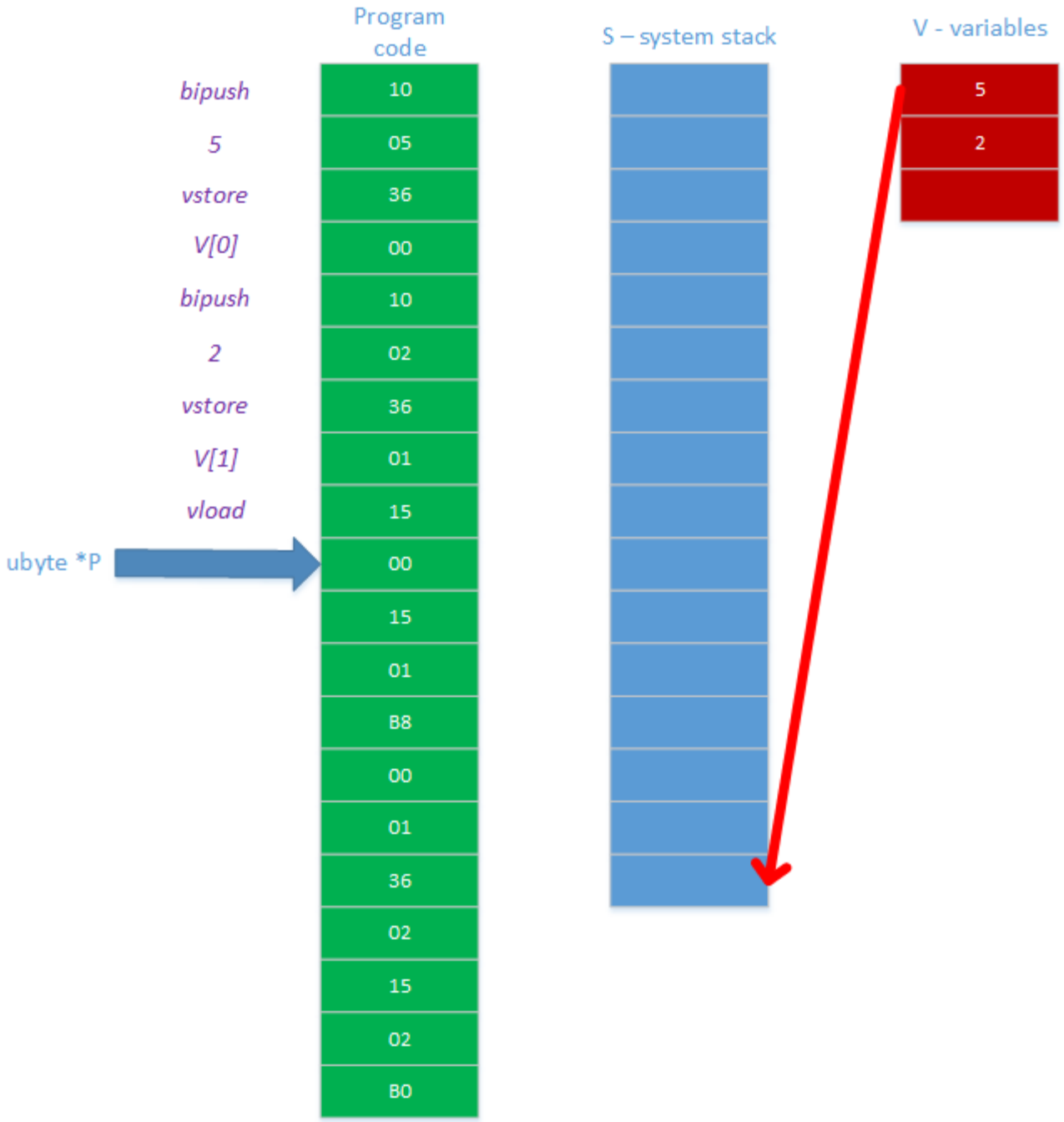

Program code
S – system stack
V - variables
bipush
5
vstore
V[0]
ubyte *P
10
05
36
00
10
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5

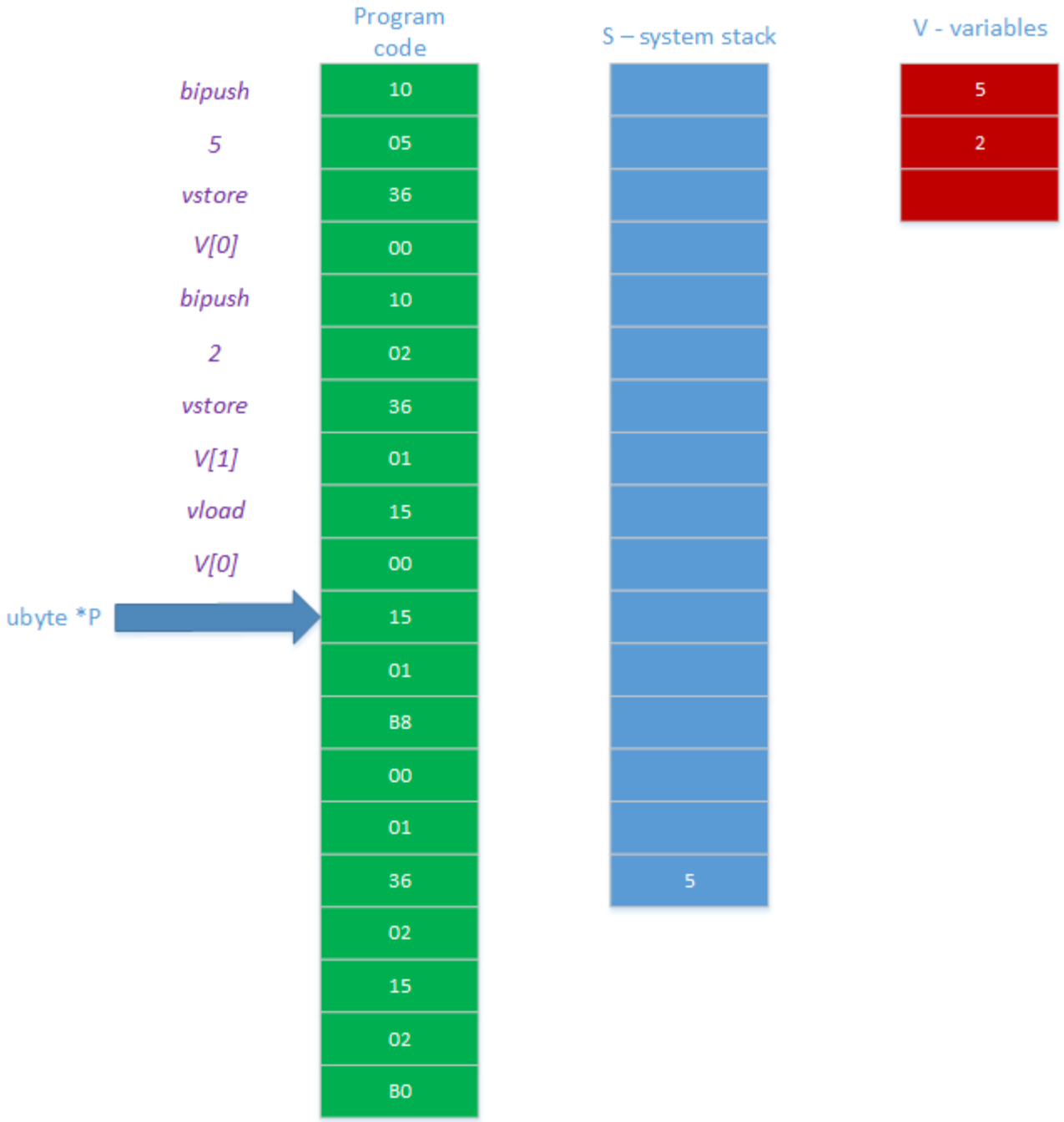

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
ubyte *P
02
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5

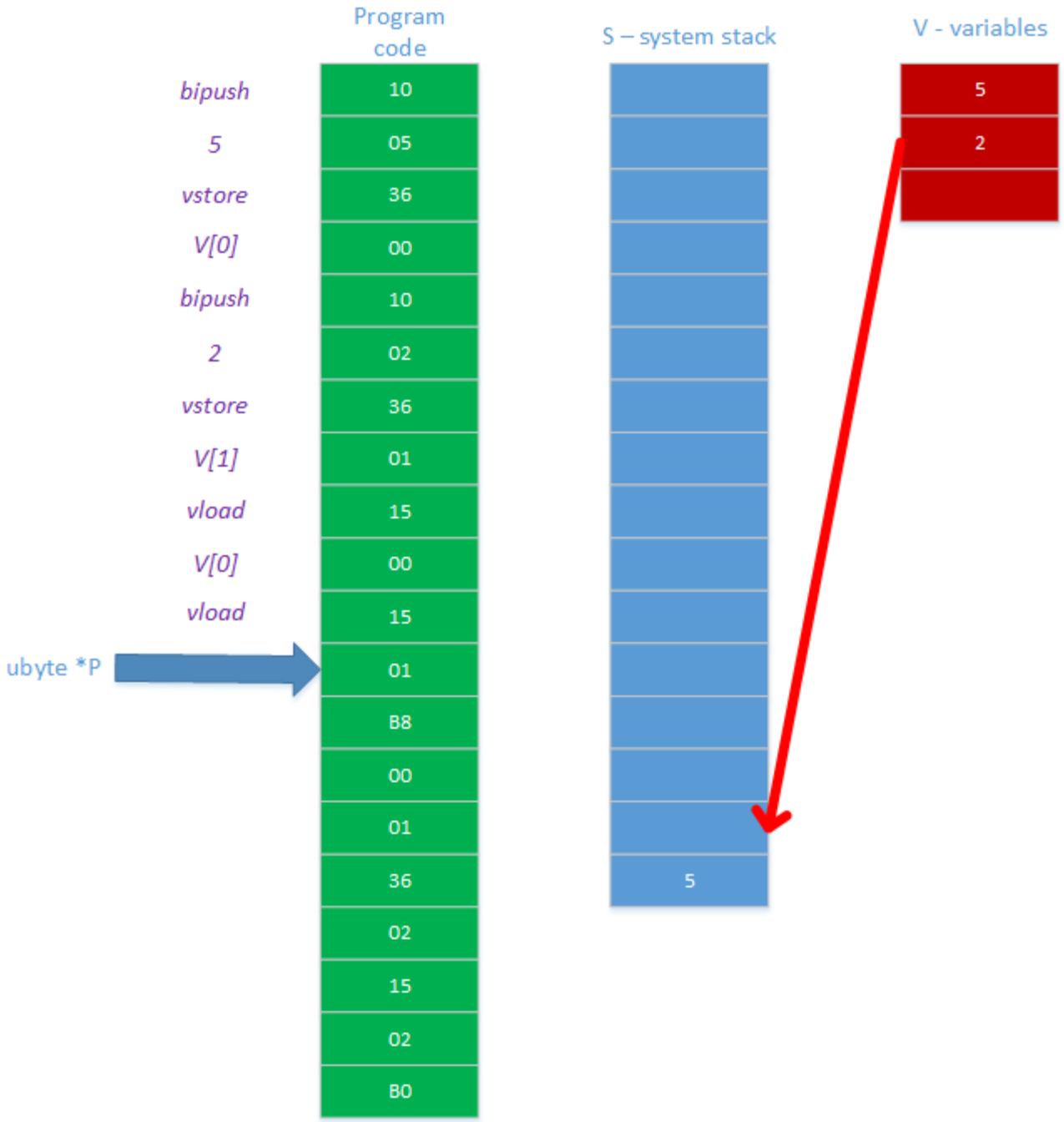

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
ubyte *P
36
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5
02

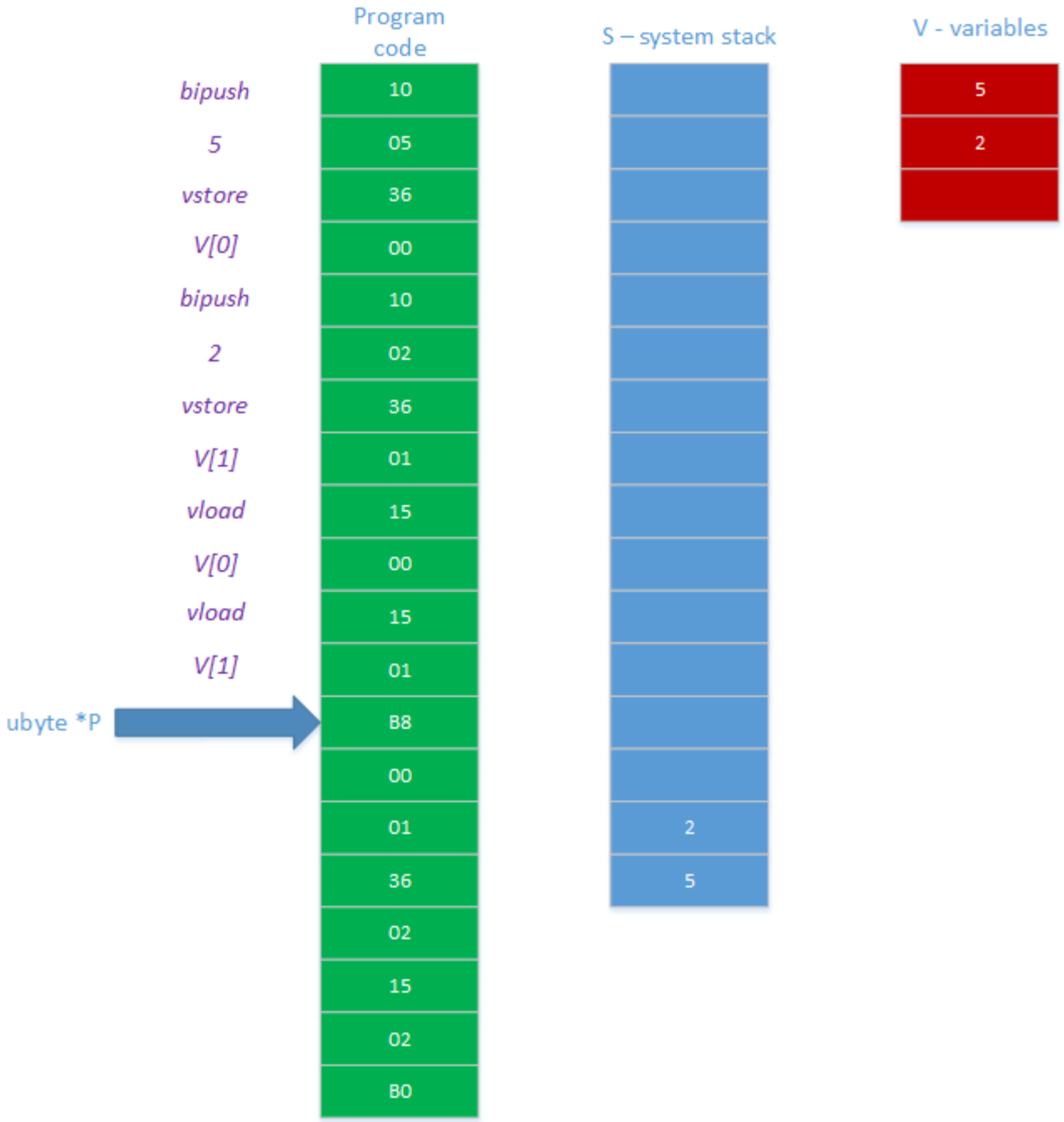

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
ubyte *P
01
15
00
15
01
B8
00
01
36
02
15
02
B0
5
02

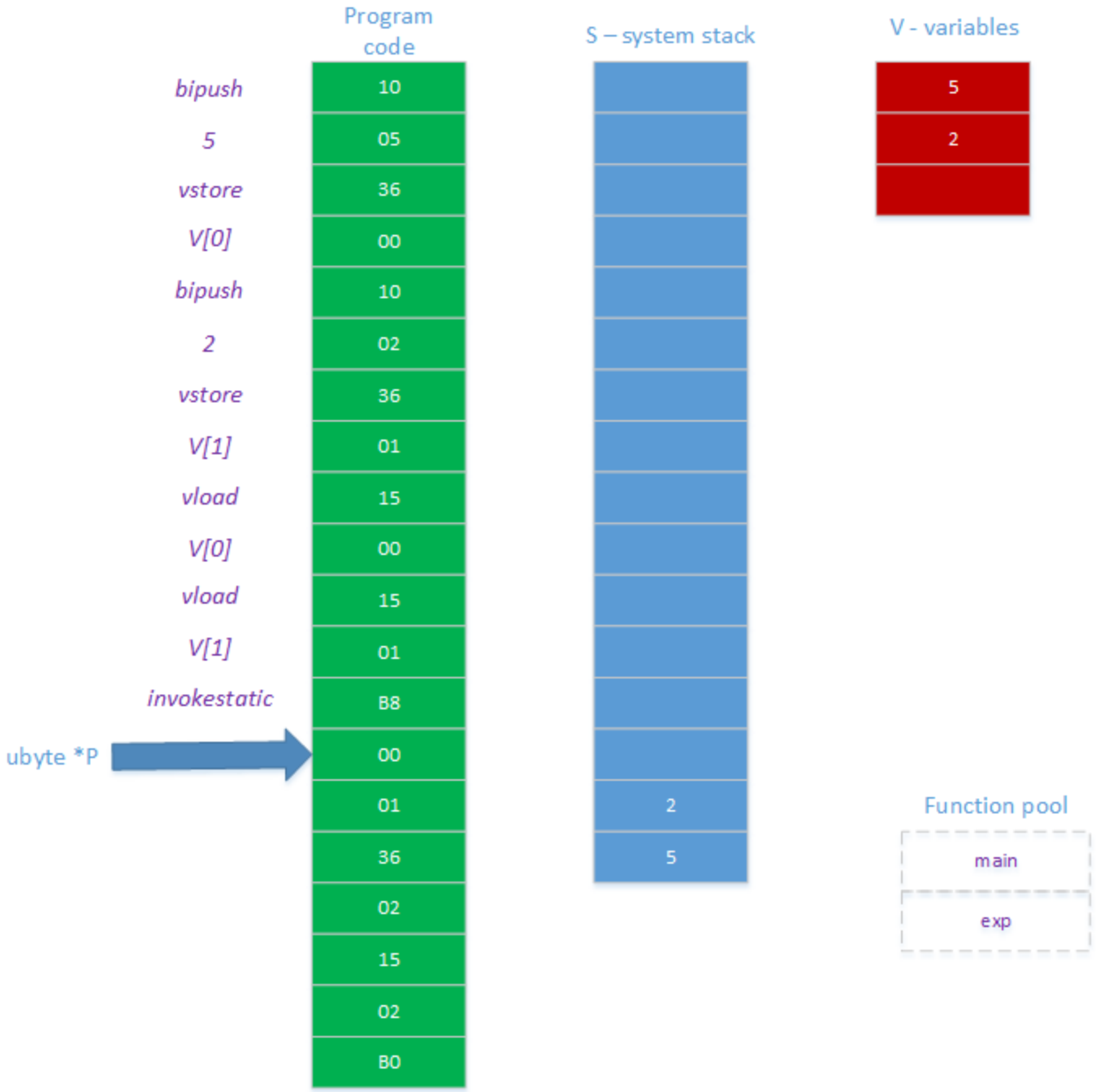

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
ubyte *P
15
00
15
01
B8
00
01
36
02
15
02
B0
5
2

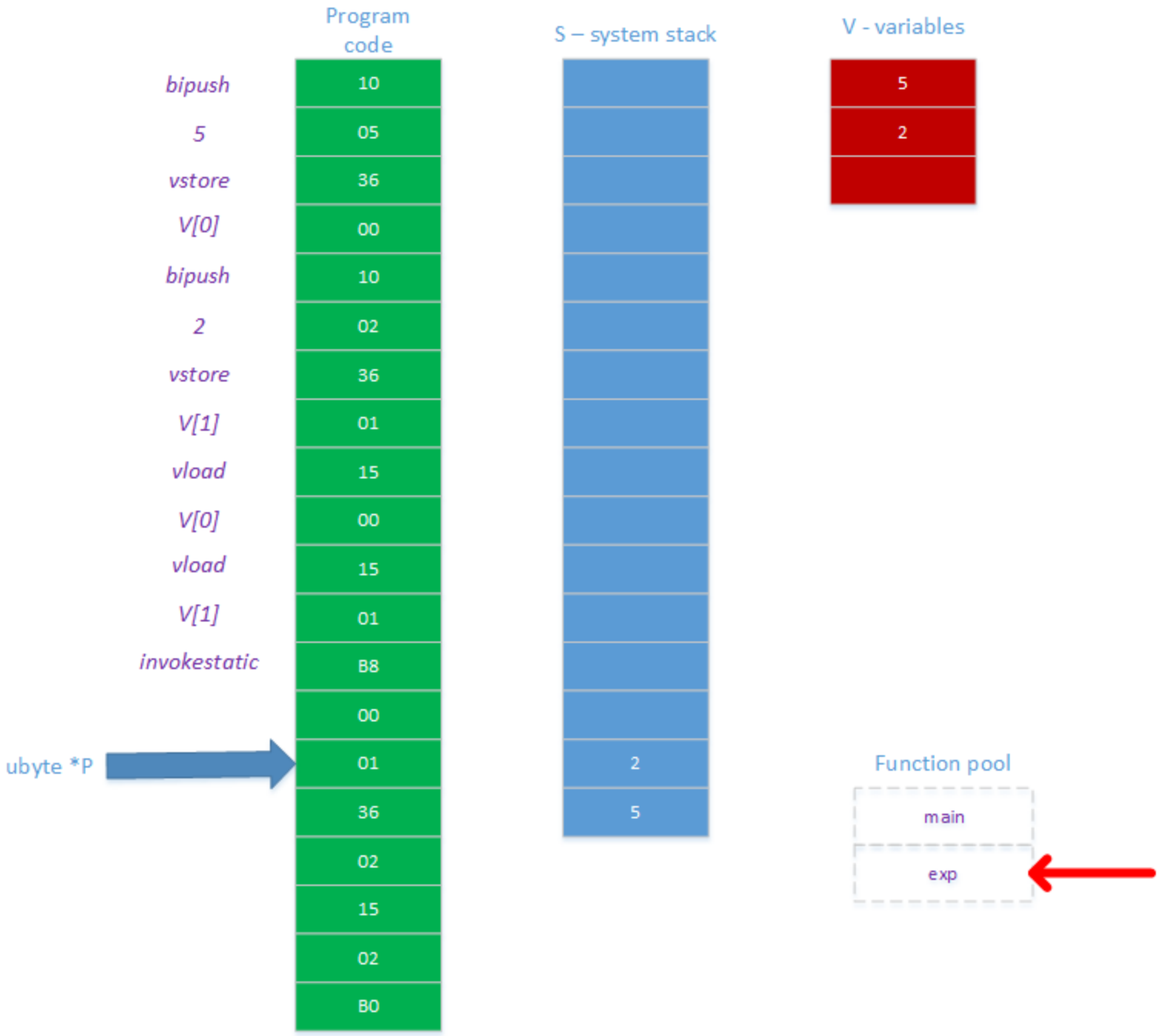

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
ubyte *P
00
15
01
B8
00
01
36
02
15
02
B0
5
2


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
ubyte *P
15
01
B8
00
01
36
02
15
02
B0
5
5
2

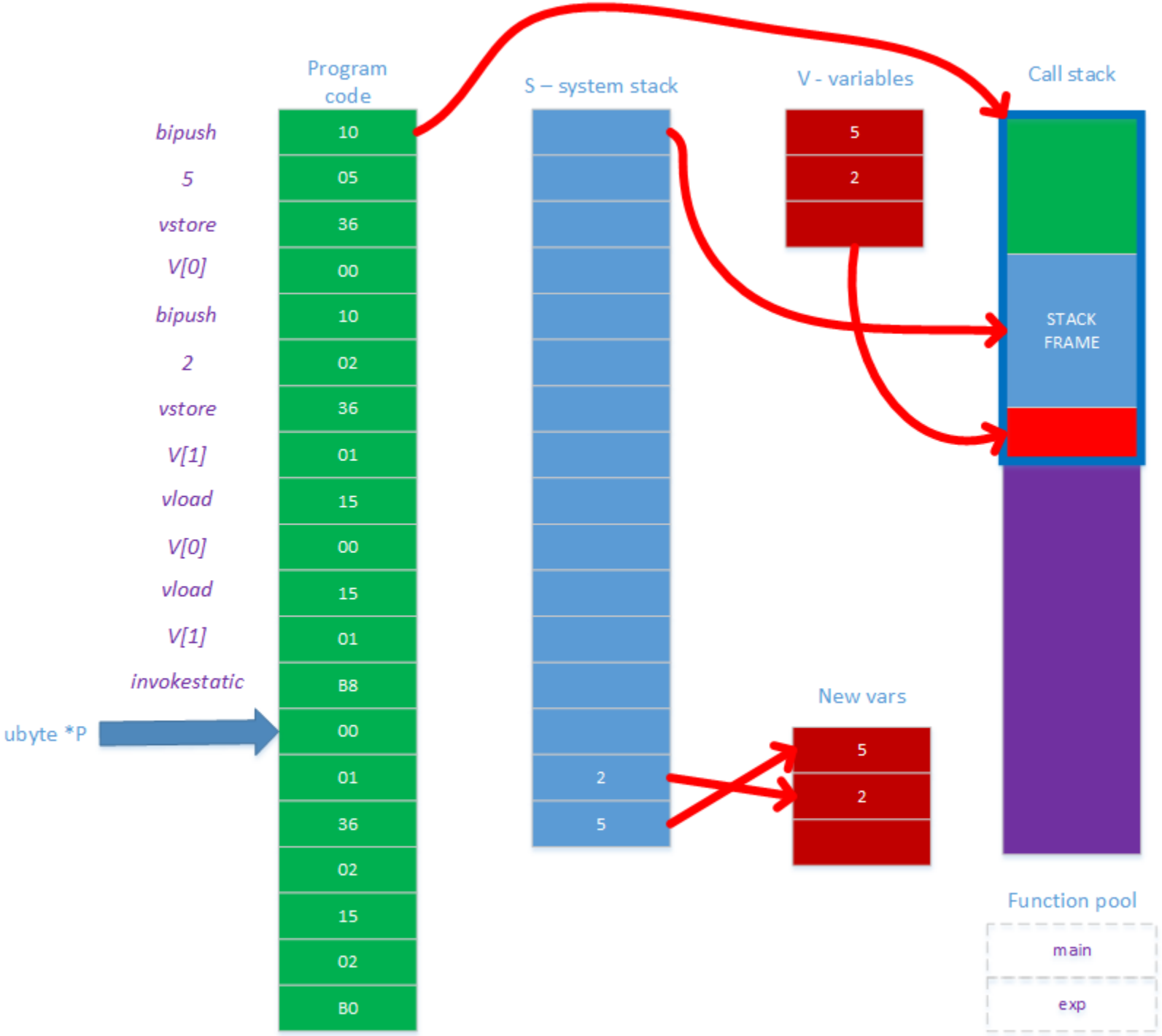

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
ubyte *P
01
B8
00
01
36
02
15
02
B0
5
5
2

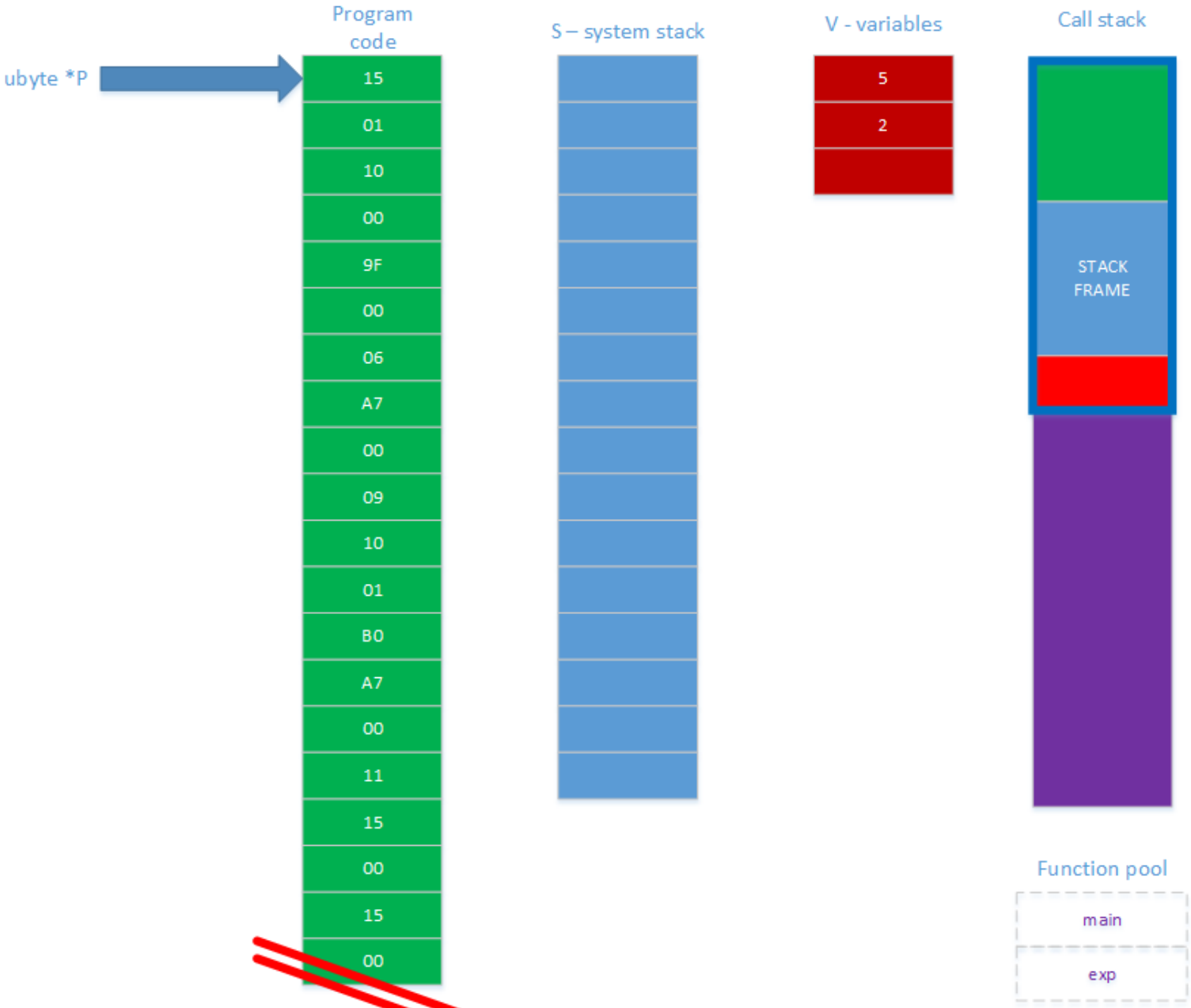

Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
ubyte *P
B8
00
01
2
36
5
02
15
02
B0
5
2


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
ubyte *P
00
01
2
36
5
02
15
02
B0
5
2
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
ubyte *P
01
36
02
15
02
B0
2
5
5
2
Function pool
main
exp

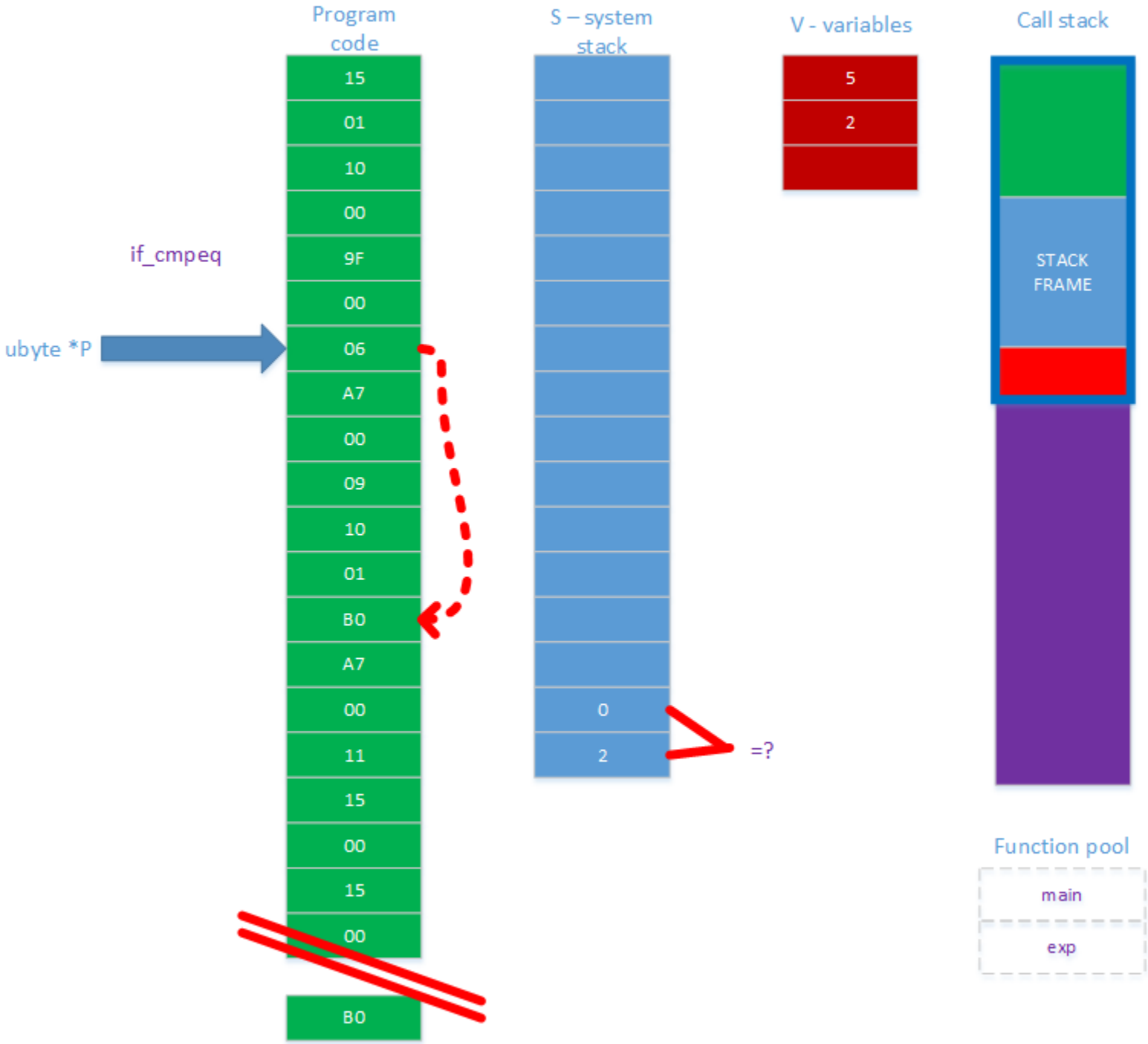
```
#<exp>
00 02              # number of arguments = 2
00 02              # number of local variables = 2
00 1E              # code length = 30 bytes
15 01    # vload 1          # e
10 00    # bipush 0         # 0
9F 00 06 # if_cmpeq +6       # if (e == 0) goto <00:then>
A7 00 09 # goto +9           # goto <01:else>
# <00:then>
10 01    # bipush 1         # 1
B0       # return           #
A7 00 11 # goto +17          # goto <02:endif>
# <01:else>
15 00    # vload 0          # b
15 00    # vload 0          # b
15 01    # vload 1          # e
10 01    # bipush 1         # 1
64       # isub             # (e - 1)
B8 00 01 # invokestatic 1   # exp(b, (e - 1))
68       # imul             # (b * exp(b, (e - 1)))
B0       # return           #
# <02:endif>
```

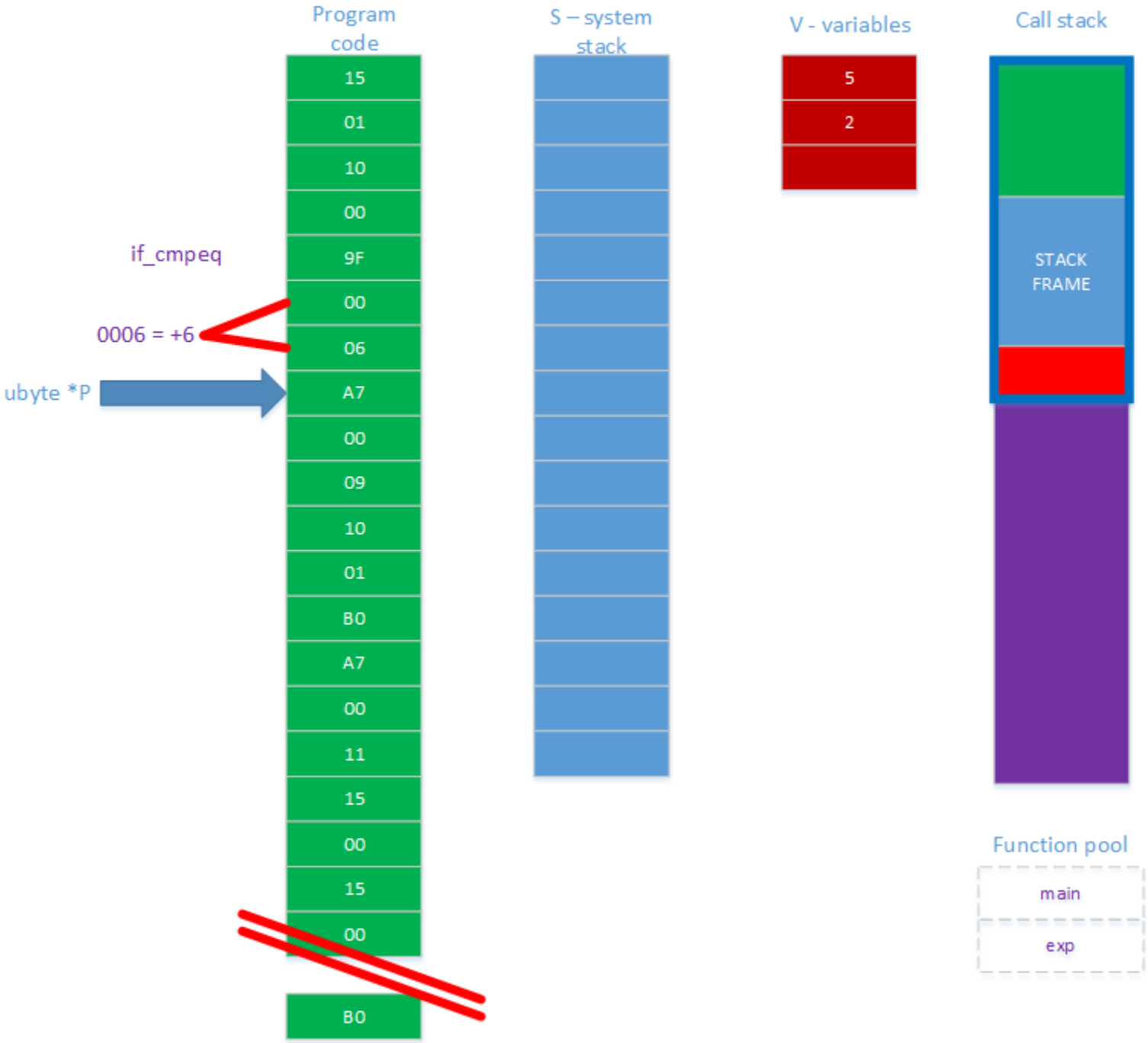

Program code
S – system stack
V - variables
Call stack
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
ubyte *P
00
01
36
02
15
02
B0
5
2
2
5
STACK FRAME
New vars
5
2
Function pool
main
exp


Program code
S – system stack
V - variables
Call stack
ubyte *P
15
01
10
00
9F
00
06
A7
00
09
10
01
B0
A7
00
11
15
00
15
00
B0
5
2
STACK FRAME
Function pool
main
exp

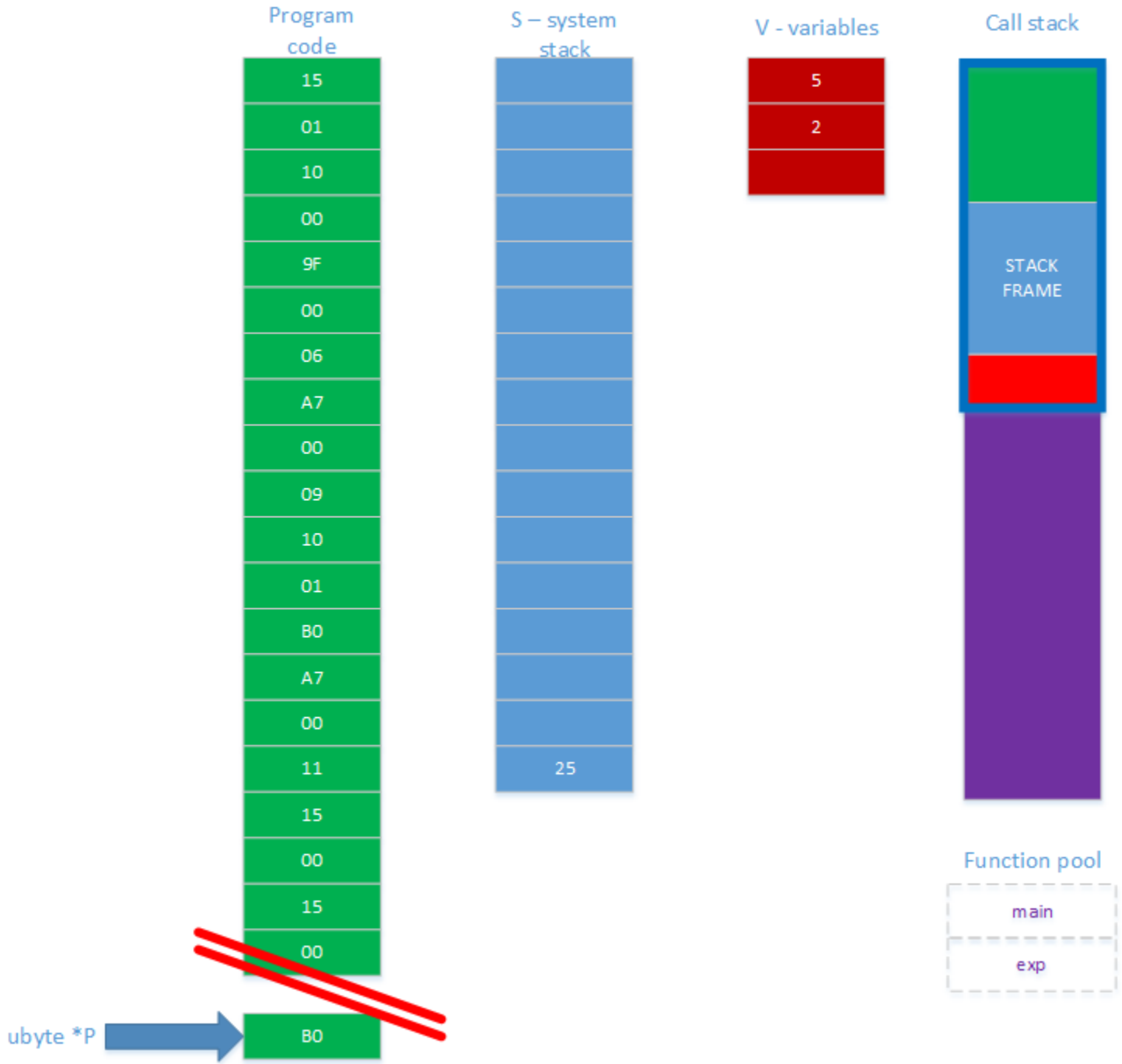
Skip ahead to the if_cmpeq

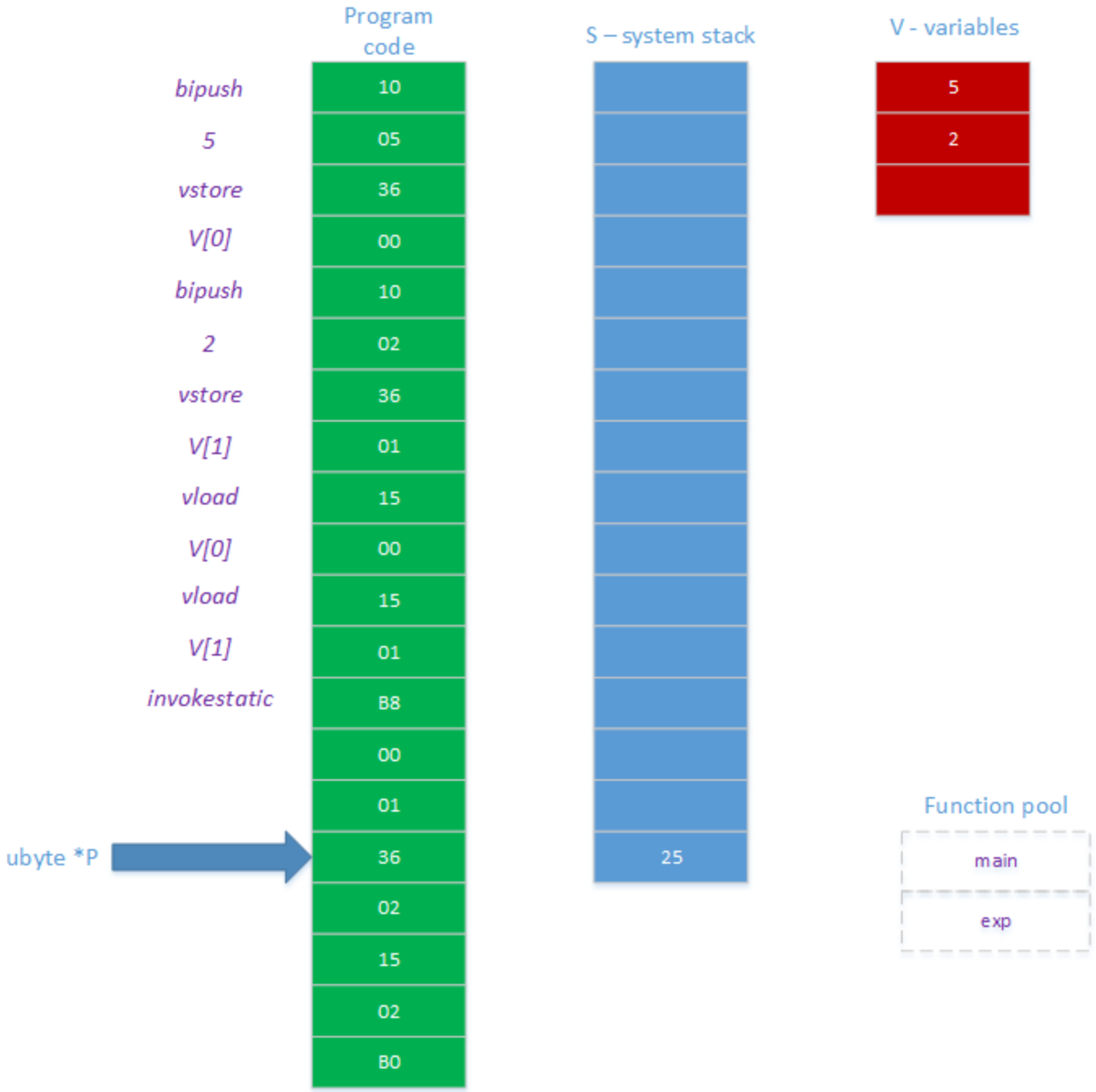

Program code
S – system stack
V - variables
Call stack
ubyte *P
15
01
10
00
9F
00
06
A7
00
09
10
01
B0
A7
00
11
15
00
15
00
B0
0
2
5
2
STACK FRAME
Function pool
main
exp

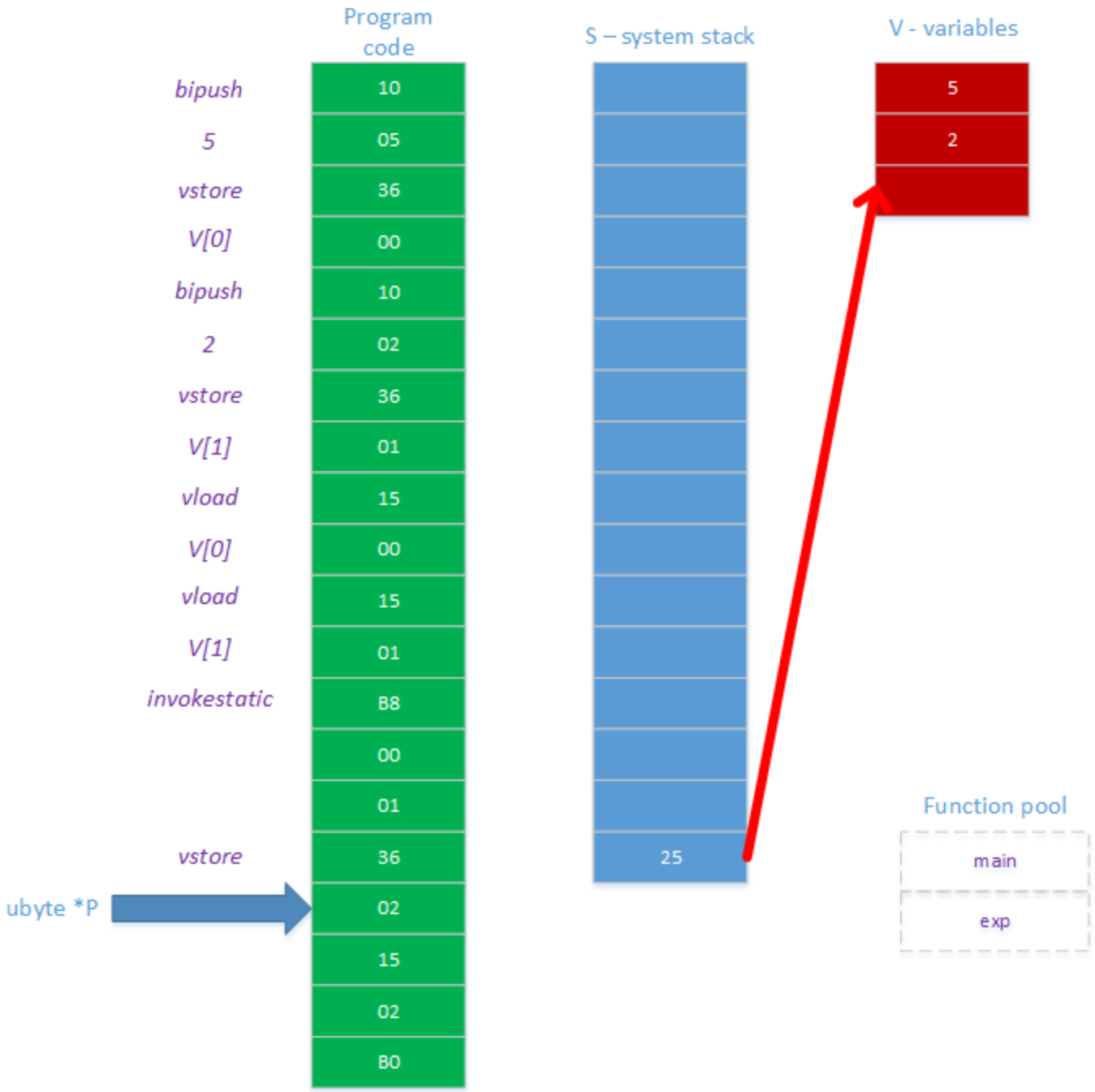

Program code
S – system stack
V - variables
Call stack
15
01
10
00
9F
00
06
A7
00
09
10
01
B0
A7
00
11
15
00
15
00
B0
if_cmpeq
ubyte *P
5
2
0
2
=?
STACK FRAME
Function pool
main
exp

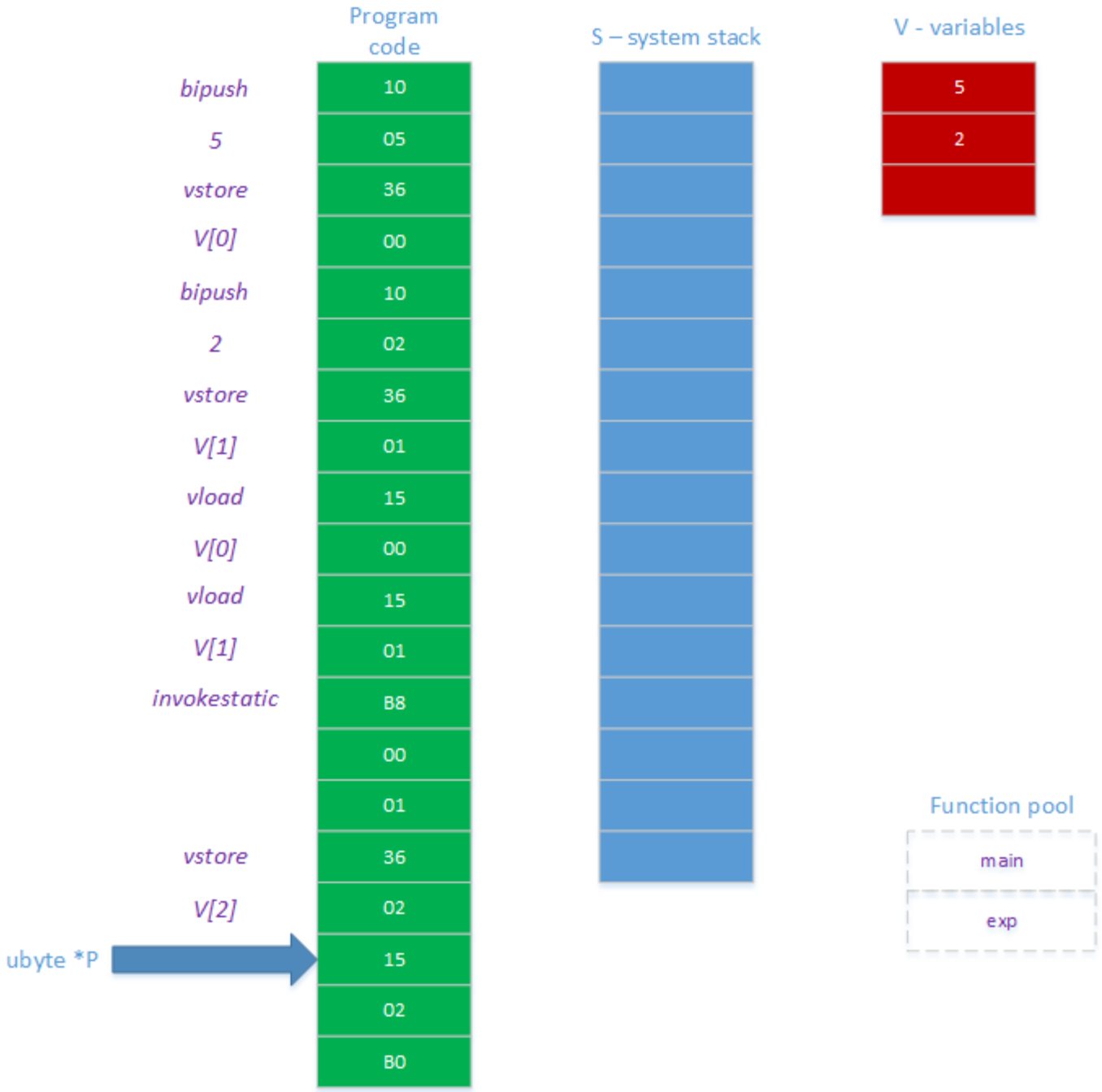

Program code
S – system stack
V - variables
Call stack
15
01
10
00
9F
00
06
A7
00
09
10
01
B0
A7
00
11
15
00
15
00
B0
5
2
STACK FRAME
if_cmpeq
0006 = +6
ubyte *P
Function pool
main
exp

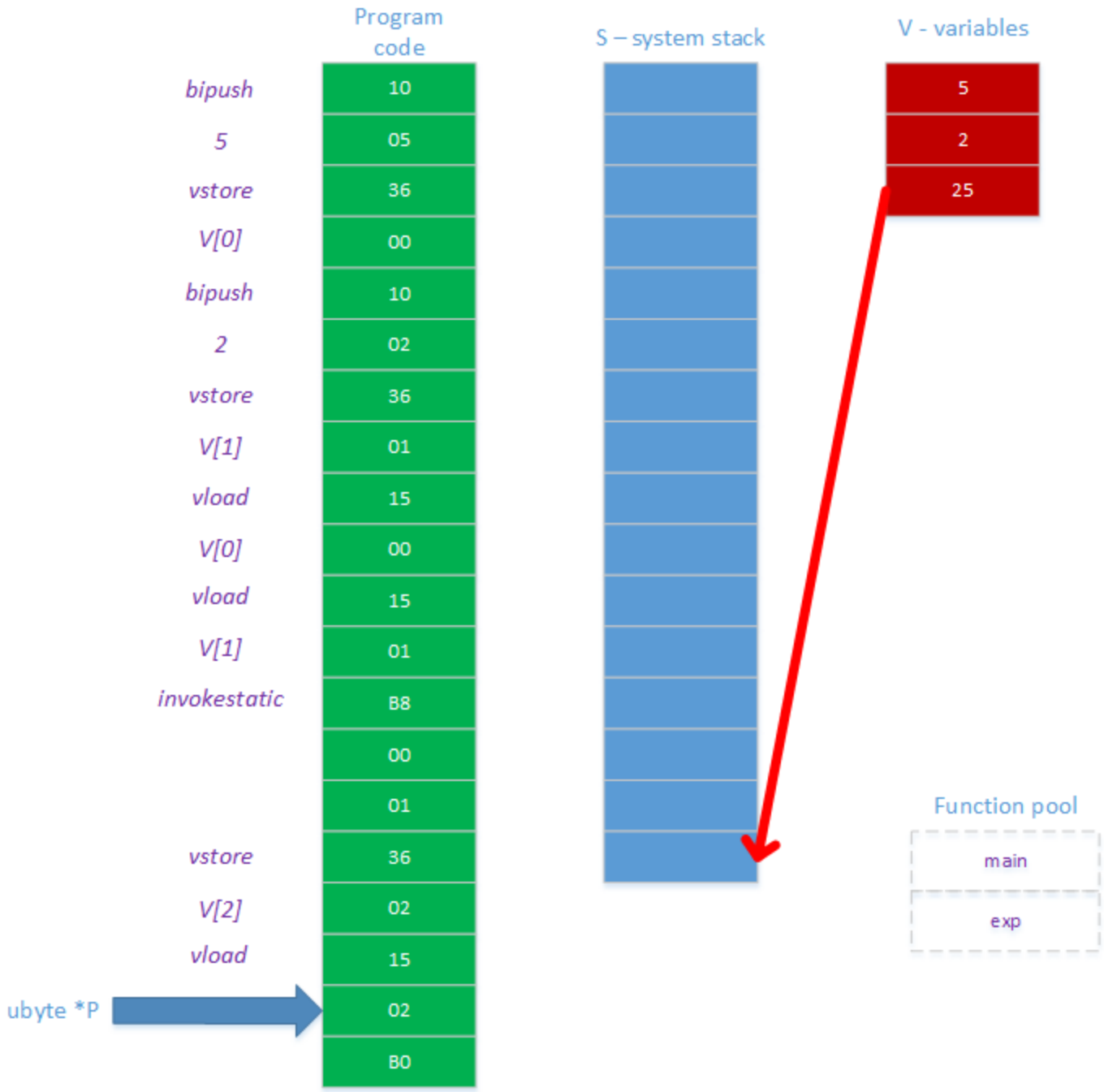
Skip ahead through function to return

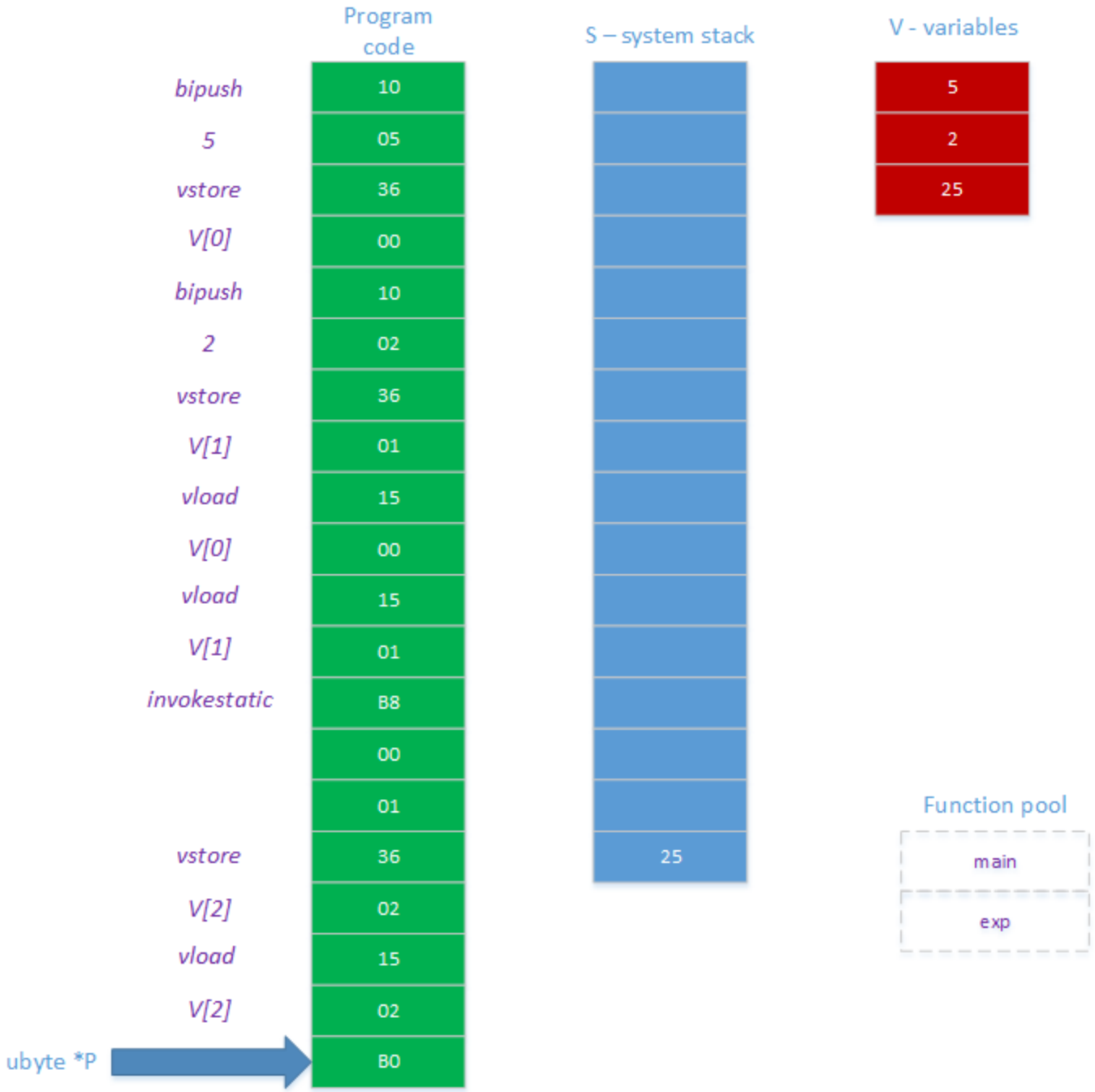

Program code
15
01
10
00
9F
00
06
A7
00
09
10
01
B0
A7
00
11
15
00
15
00
B0
ubyte *P
S – system stack
25
V - variables
5
2
Call stack
STACK FRAME
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
01
ubyte *P
36
02
15
02
B0
5
2
25
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
01
vstore
36
ubyte *P
02
15
02
B0
25
5
2
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
01
vstore
36
V[2]
02
ubyte *P
15
02
B0
5
2
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
01
vstore
36
V[2]
02
vload
15
ubyte *P
02
B0
5
2
25
Function pool
main
exp


Program code
S – system stack
V - variables
bipush
10
5
05
vstore
36
V[0]
00
bipush
10
2
02
vstore
36
V[1]
01
vload
15
V[0]
00
vload
15
V[1]
01
invokestatic
B8
00
01
vstore
36
25
V[2]
02
vload
15
V[2]
02
ubyte *P
B0
5
2
25
Function pool
main
exp

Call stack is empty, so pop 25 from stack and return