UNIVERSITY OF CENTRAL FLORIDA

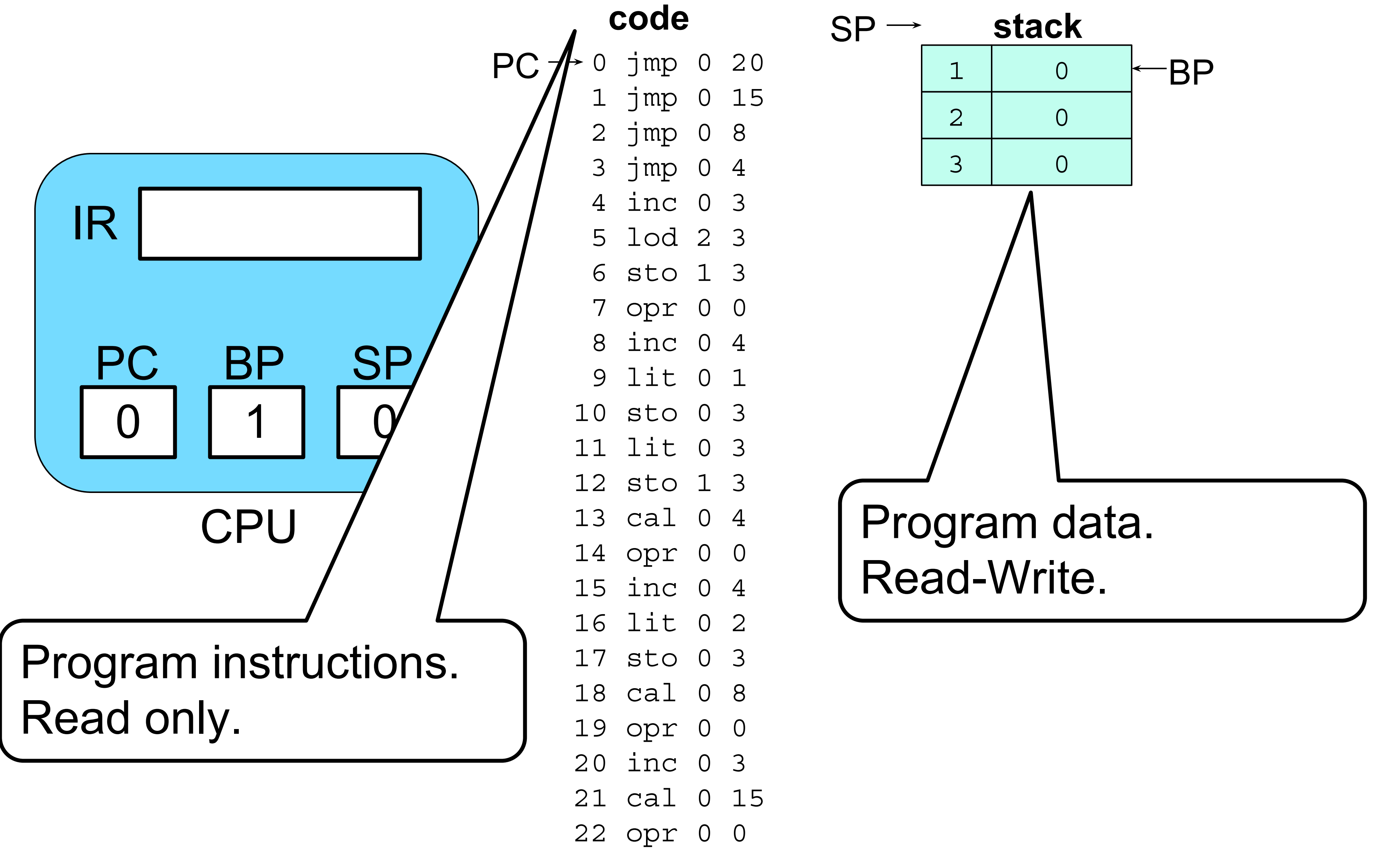
Lab 3
PM/0 Code Execution

COP 3402
(September 12th, 2018)

Important notes

- This PM/0 example doesn’t match the PM/0 that you must implement for HW1.
- Opcodes, instruction names and activation record may be different.
- When in doubt, follow the assignment or ask a TA.

P-machine



code

```
 0 jmp 0 20
 1 jmp 0 15
 2 jmp 0 8
 3 jmp 0 4
 4 inc 0 3
 5 lod 2 3
 6 sto 1 3
 7 opr 0 0
 8 inc 0 4
 9 lit 0 1
10 sto 0 3
11 lit 0 3
12 sto 1 3
13 cal 0 4
14 opr 0 0
15 inc 0 4
16 lit 0 2
17 sto 0 3
18 cal 0 8
19 opr 0 0
20 inc 0 3
21 cal 0 15
22 opr 0 0
```

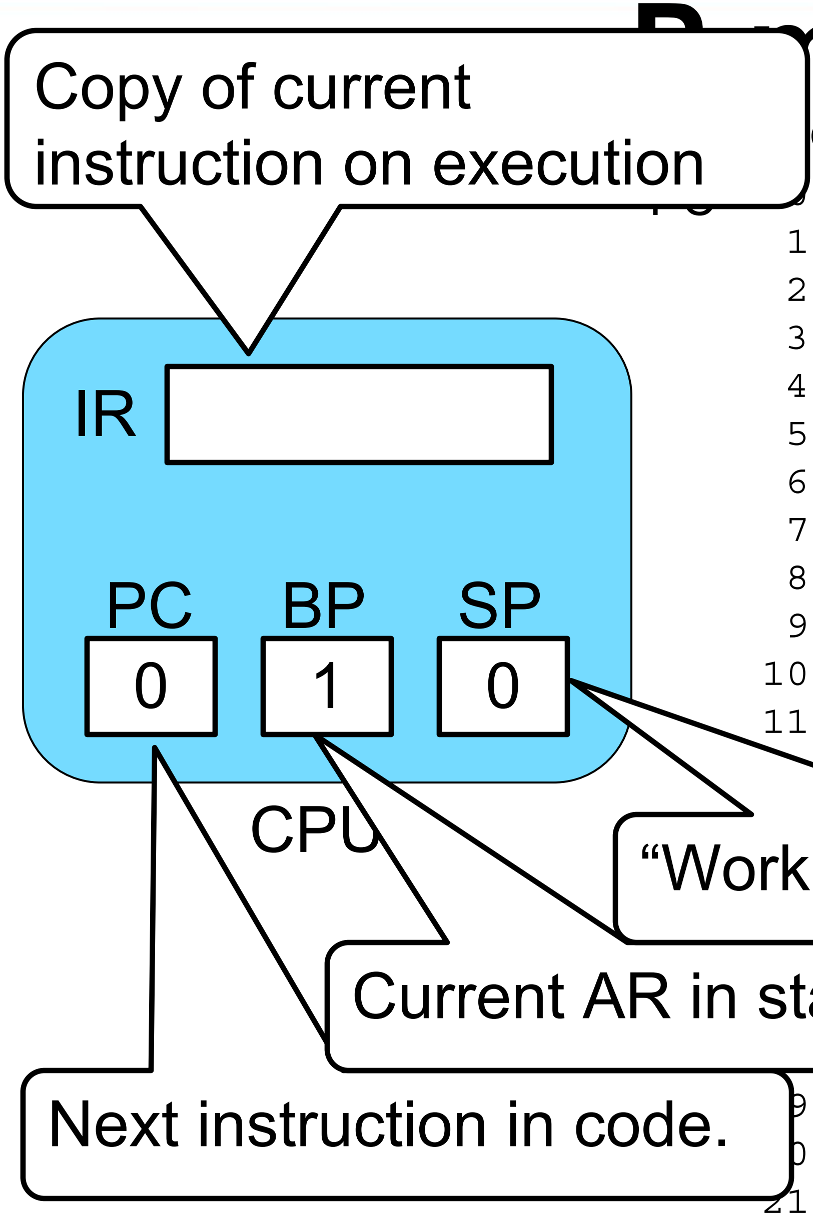
stack

1	0
2	0
3	0

Program instructions.
Read only.

Program data.
Read-Write.

P-machine


Copy of current instruction on execution
IR
PC
0
BP
1
SP
0
CPU
"Working place" in stack.
Current AR in stack.
Next instruction in code.

code

```
 0  jmp  0  20
 1  jmp  0  15
 2  jmp  0  8
 3  jmp  0  4
 4  inc  0  3
 5  lod  2  3
 6  sto  1  3
 7  opr  0  0
 8  inc  0  4
 9  lit  0  1
10  sto  0  3
11  lit  0  3
    sto  1  3
19  opr  0  0
20  inc  0  3
21  cal  0  15
22  opr  0  0
```

SP →

stack

1	0
2	0
3	0

← BP

P-machine ISA

opcode

↓

02 - OPR:

RTN	**0,0**	→ **Return operation**	(i.e. return from subroutine)
OPR	**0,1**	→ **NEG**	(- stack[sp])
OPR	**0,2**	→ **ADD**	(sp ← sp – 1 and stack[sp] ← stack[sp] + stack[sp + 1])
OPR	**0,3**	→ **SUB**	(sp ← sp – 1 and stack[sp] ← stack[sp] - stack[sp + 1])
OPR	**0,4**	→ **MUL**	(sp ← sp – 1 and stack[sp] ← stack[sp] * stack[sp + 1])
OPR	**0,5**	→ **DIV**	(sp ← sp – 1 and stack[sp] ← stack[sp] div stack[sp + 1])
OPR	**0,6**	→ **ODD**	(stack[sp] ← stack[sp] mod 2) or ord(odd(stack[sp]))
OPR	**0,7**	→ **MOD**	(sp ← sp – 1 and stack[sp] ← stack[sp] mod stack[sp + 1])
OPR	**0,8**	→ **EQL**	(sp ← sp – 1 and stack[sp] ← stack[sp] = =stack[sp + 1])
OPR	**0,9**	→ **NEQ**	(sp ← sp – 1 and stack[sp] ← stack[sp] != stack[sp + 1])
OPR	**0,10**	→ **LSS**	(sp ← sp – 1 and stack[sp] ← stack[sp] < stack[sp + 1])
OPR	**0,11**	→ **LEQ**	(sp ← sp – 1 and stack[sp] ← stack[sp] <= stack[sp + 1])
OPR	**0,12**	→ **GTR**	(sp ← sp – 1 and stack[sp] ← stack[sp] > stack[sp + 1])
OPR	**0,13**	→ **GEQ**	(sp ← sp – 1 and stack[sp] ← stack[sp] >= stack[sp + 1])

P-machine ISA

opcode

```
01 -  LIT   0, M →  sp ← sp +1;
                 stack[sp] ← M;

02 – RTN   0, 0 →   sp ← bp -1;
                 pc ← stack[sp + 3];
                 bp ← stack[sp + 2];

03 – LOD   L, M → sp ← sp +1;
                 stack[sp] ← stack[ base(L,BP) + M];

04 – STO   L, M →  stack[ base(L,BP) + M] ← stack[sp];
                 sp ← sp -1;
```

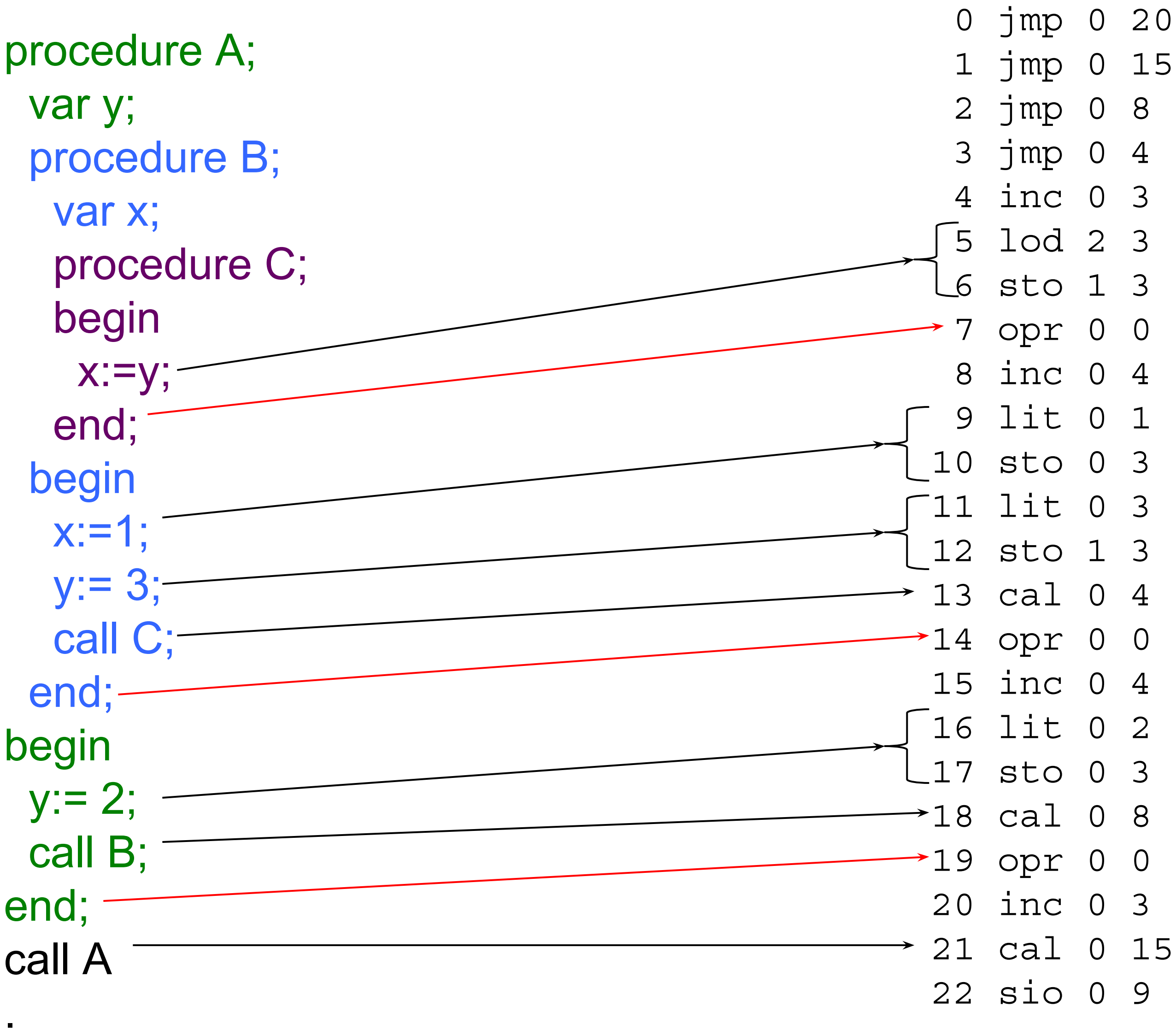
P-machine ISA

opcode

↓

```
05 - CAL   L, M →  stack[sp + 1]  ←  base(L, bp);  /* static link (SL)
                   stack[sp + 2]  ← bp;           /*  dynamic link (DL)
                   stack[sp + 3]  ← pc            /*  return address (RA)
                   bp ← sp + 1;
                   pc ← M;

06 – INC   0, M →  sp ← sp + M;


07 – JMP  0, M →   pc = M;


08 – JPC  0, M →   if stack[sp] == 0 then { pc ← M;
                          sp ← sp - 1;
                          }

09 – SIO  0, 0  →   print (stack[sp]);
               sp ← sp – 1;
```

Nested Code



```
procedure A;
  var y;
  procedure B;
    var x;
    procedure C;
    begin
      x:=y;
    end;
  begin
    x:=1;
    y:= 3;
    call C;
  end;
begin
  y:= 2;
  call B;
end;
call A
.
```

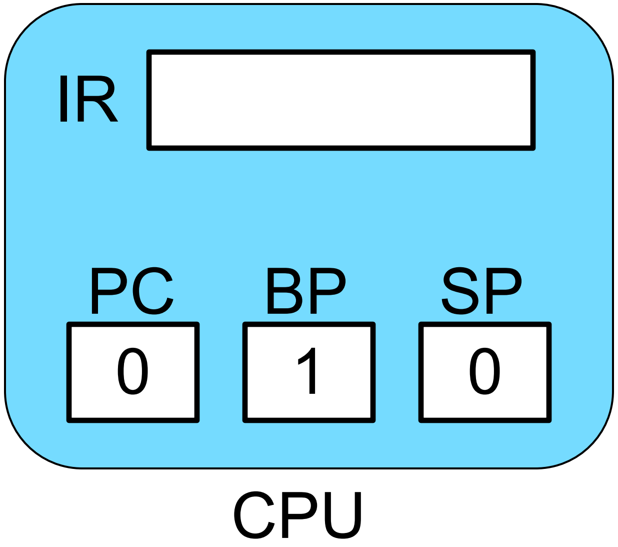
code

```
 0 jmp 0 20
 1 jmp 0 15
 2 jmp 0 8
 3 jmp 0 4
 4 inc 0 3
 5 lod 2 3
 6 sto 1 3
 7 opr 0 0
 8 inc 0 4
 9 lit 0 1
10 sto 0 3
11 lit 0 3
12 sto 1 3
13 cal 0 4
14 opr 0 0
15 inc 0 4
16 lit 0 2
17 sto 0 3
18 cal 0 8
19 opr 0 0
20 inc 0 3
21 cal 0 15
22 sio 0 9
```

```
RTN 0,0 →
sp ← bp -1;
pc ← stack[sp + 3];
bp ← stack[sp + 2];
```

Running Nested Code on PM/0

Initial State



IR

PC	BP	SP
0	1	0

CPU

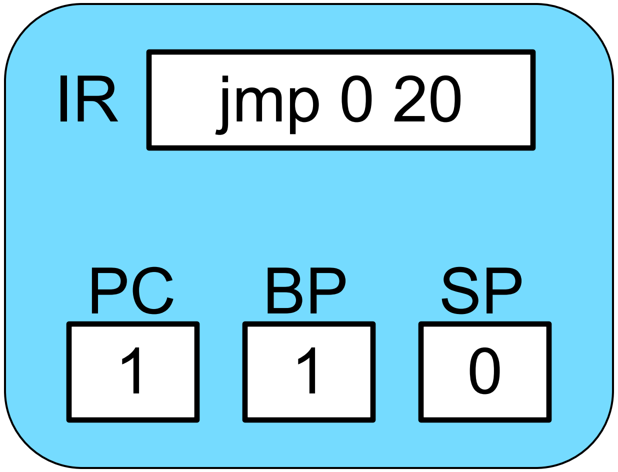
code

```
PC → 0  jmp 0 20
     1  jmp 0 15
     2  jmp 0 8
     3  jmp 0 4
     4  inc 0 3
     5  lod 2 3
     6  sto 1 3
     7  opr 0 0
     8  inc 0 4
     9  lit 0 1
    10  sto 0 3
    11  lit 0 3
    12  sto 1 3
    13  cal 0 4
    14  opr 0 0
    15  inc 0 4
    16  lit 0 2
    17  sto 0 3
    18  cal 0 8
    19  opr 0 0
    20  inc 0 3
    21  cal 0 15
    22  opr 0 0
```

SP → **stack**

1	0	← BP
2	0	
3	0	

After FETCH, no execution yet!



IR: jmp 0 20

PC: 1

BP: 1

SP: 0

CPU

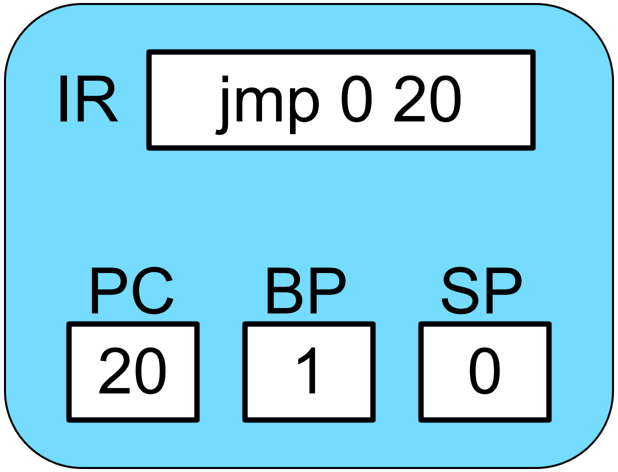
code

```
      0  jmp  0  20
PC →  1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

SP → **stack**

1	0	← BP
2	0	
3	0	

After EXECUTION



CPU

code

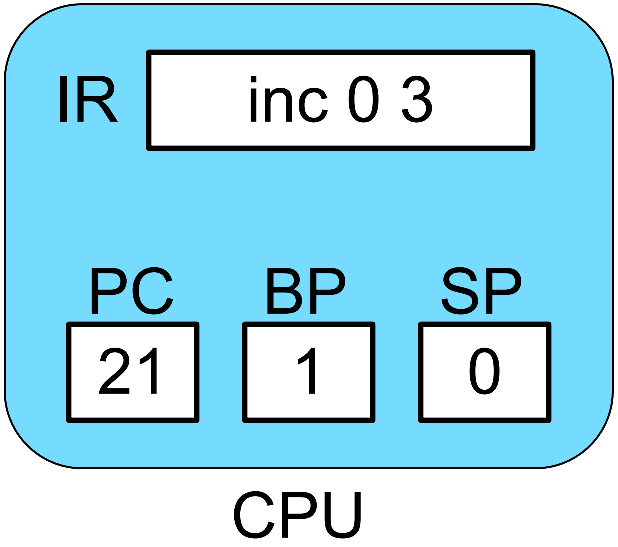
```
       0  jmp  0  20
       1  jmp  0  15
       2  jmp  0  8
       3  jmp  0  4
       4  inc  0  3
       5  lod  2  3
       6  sto  1  3
       7  opr  0  0
       8  inc  0  4
       9  lit  0  1
      10  sto  0  3
      11  lit  0  3
      12  sto  1  3
      13  cal  0  4
      14  opr  0  0
      15  inc  0  4
      16  lit  0  2
      17  sto  0  3
      18  cal  0  8
      19  opr  0  0
PC → 20  inc  0  3
      21  cal  0  15
      22  opr  0  0
```

SP →

stack

1	0	← BP
2	0	
3	0	

After FETCH



IR inc 0 3

PC 21

BP 1

SP 0

CPU

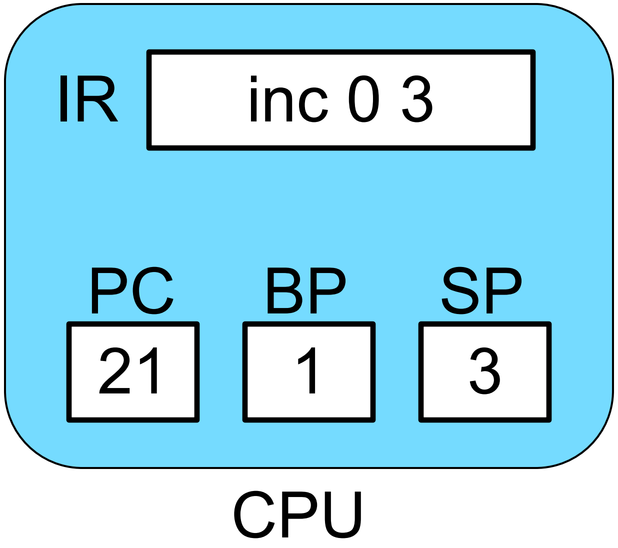
code

```
    0  jmp  0  20
    1  jmp  0  15
    2  jmp  0  8
    3  jmp  0  4
    4  inc  0  3
    5  lod  2  3
    6  sto  1  3
    7  opr  0  0
    8  inc  0  4
    9  lit  0  1
   10  sto  0  3
   11  lit  0  3
   12  sto  1  3
   13  cal  0  4
   14  opr  0  0
   15  inc  0  4
   16  lit  0  2
   17  sto  0  3
   18  cal  0  8
   19  opr  0  0
   20  inc  0  3
PC →21  cal  0  15
   22  opr  0  0
```

SP → **stack**

1	0	← BP
2	0	
3	0	

After EXECUTION



IR inc 0 3

PC 21

BP 1

SP 3

CPU

code

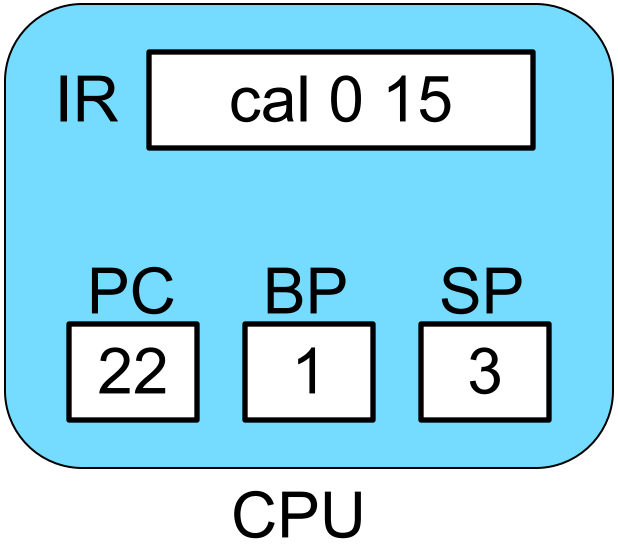
```
    0  jmp  0  20
    1  jmp  0  15
    2  jmp  0  8
    3  jmp  0  4
    4  inc  0  3
    5  lod  2  3
    6  sto  1  3
    7  opr  0  0
    8  inc  0  4
    9  lit  0  1
   10  sto  0  3
   11  lit  0  3
   12  sto  1  3
   13  cal  0  4
   14  opr  0  0
   15  inc  0  4
   16  lit  0  2
   17  sto  0  3
   18  cal  0  8
   19  opr  0  0
   20  inc  0  3
PC →21  cal  0  15
   22  opr  0  0
```

stack

	1	0	← BP
	2	0	
SP →	3	0	

We “reserve space” for some data by incrementing the SP.

After FETCH


IR

cal 0 15

PC

22

BP

1

SP

3

CPU

code

```
     0  jmp  0  20
     1  jmp  0  15
     2  jmp  0  8
     3  jmp  0  4
     4  inc  0  3
     5  lod  2  3
     6  sto  1  3
     7  opr  0  0
     8  inc  0  4
     9  lit  0  1
    10  sto  0  3
    11  lit  0  3
    12  sto  1  3
    13  cal  0  4
    14  opr  0  0
    15  inc  0  4
    16  lit  0  2
    17  sto  0  3
    18  cal  0  8
    19  opr  0  0
    20  inc  0  3
    21  cal  0  15
PC →22  opr  0  0
```

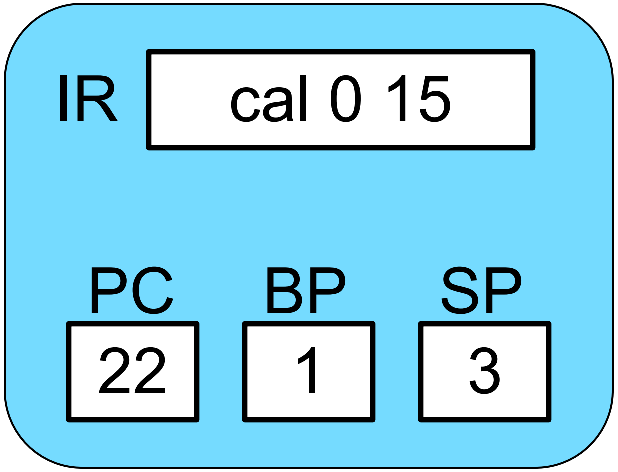
stack

	1	0	← BP
	2	0	
SP →	3	0	

CAL L, M →

```
stack[sp + 1] ← base(L);
     stack[sp + 2] ← bp;
     stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

Executing CAL



CPU

code

```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
PC → 22  opr  0  0
```

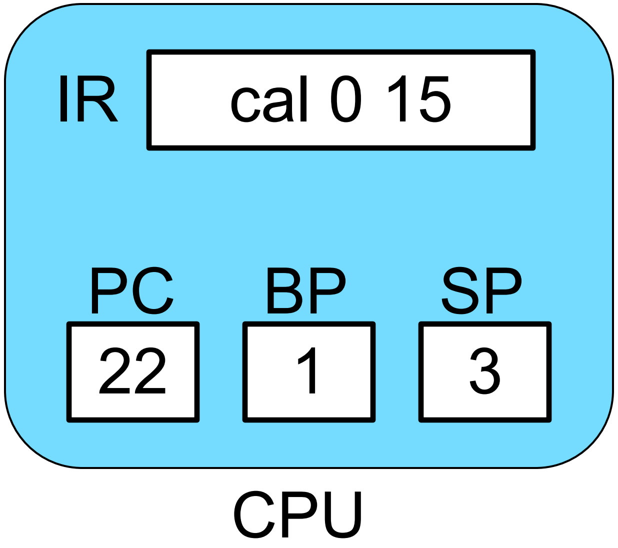
stack

	1	0	← BP
	2	0	
SP →	3	0	
	4	1	

CAL L, M →

```
stack[sp + 1] ← base(L);
        stack[sp + 2] ← bp;
        stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

Executing CAL



IR: cal 0 15

PC	BP	SP
22	1	3

CPU

code

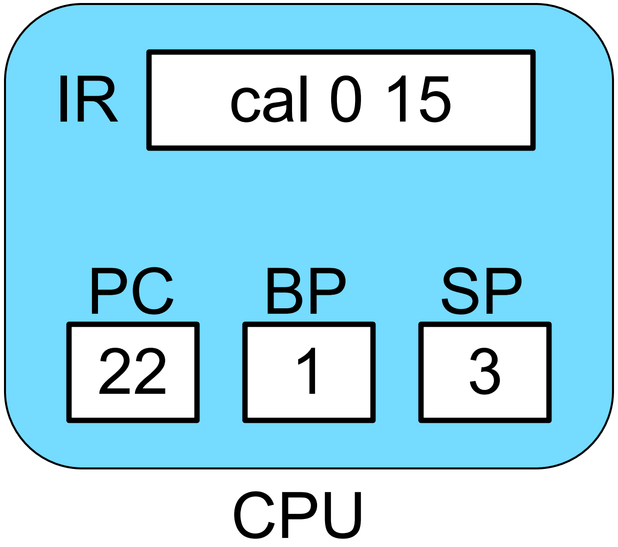
```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
PC →22  opr  0  0
```

stack

	1	0	← BP
	2	0	
SP →	3	0	
	4	1	
	5	1	

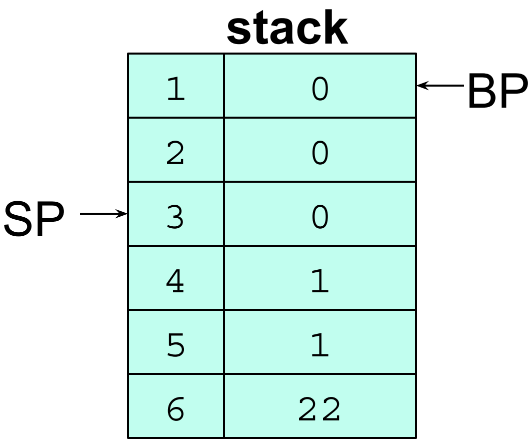
```
CAL   L, M →

stack[sp + 1] ← base(L);
        stack[sp + 2] ← bp;
        stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

Executing CAL



code

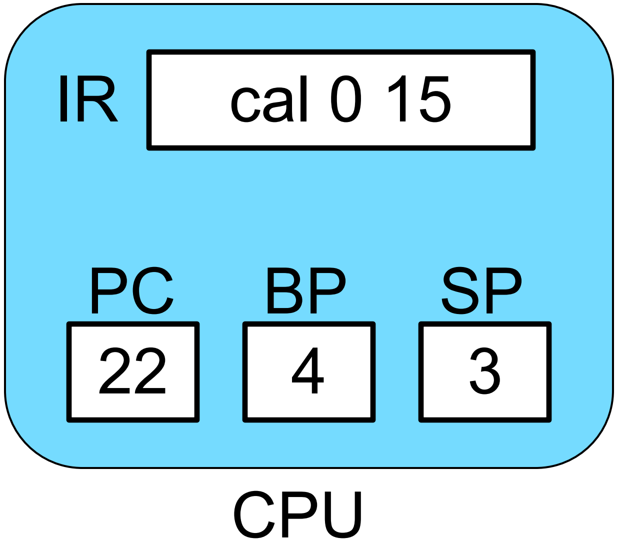
```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
PC → 22  opr  0  0
```



1	0
2	0
3	0
4	1
5	1
6	22

```
CAL   L, M →

stack[sp + 1] ← base(L);
      stack[sp + 2] ← bp;
      stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

Executing CAL



code

```
     0 jmp 0 20
     1 jmp 0 15
     2 jmp 0 8
     3 jmp 0 4
     4 inc 0 3
     5 lod 2 3
     6 sto 1 3
     7 opr 0 0
     8 inc 0 4
     9 lit 0 1
    10 sto 0 3
    11 lit 0 3
    12 sto 1 3
    13 cal 0 4
    14 opr 0 0
    15 inc 0 4
    16 lit 0 2
    17 sto 0 3
    18 cal 0 8
    19 opr 0 0
    20 inc 0 3
    21 cal 0 15
PC →22 opr 0 0
```

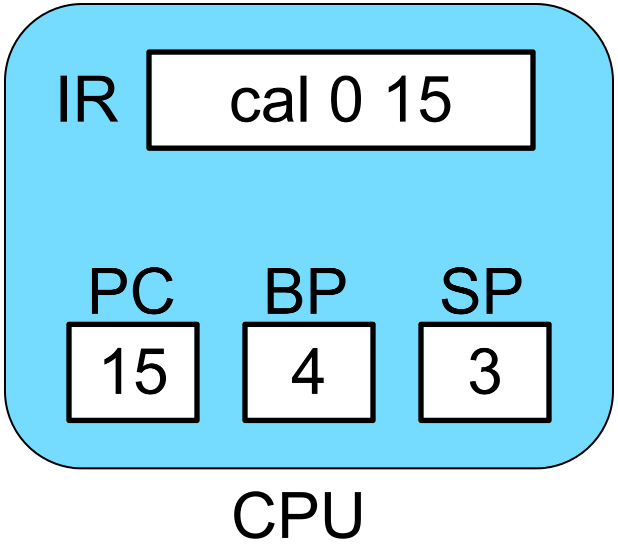
stack

	1	0	
	2	0	
SP →	3	0	
	4	1	← BP
	5	1	
	6	22	

CAL L, M →

```
stack[sp + 1] ← base(L);
        stack[sp + 2] ← bp;
        stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

Executing CAL



IR cal 0 15

PC 15

BP 4

SP 3

CPU

code

```
   0  jmp  0  20
   1  jmp  0  15
   2  jmp  0  8
   3  jmp  0  4
   4  inc  0  3
   5  lod  2  3
   6  sto  1  3
   7  opr  0  0
   8  inc  0  4
   9  lit  0  1
  10  sto  0  3
  11  lit  0  3
  12  sto  1  3
  13  cal  0  4
  14  opr  0  0
PC → 15  inc  0  4
  16  lit  0  2
  17  sto  0  3
  18  cal  0  8
  19  opr  0  0
  20  inc  0  3
  21  cal  0  15
  22  opr  0  0
```

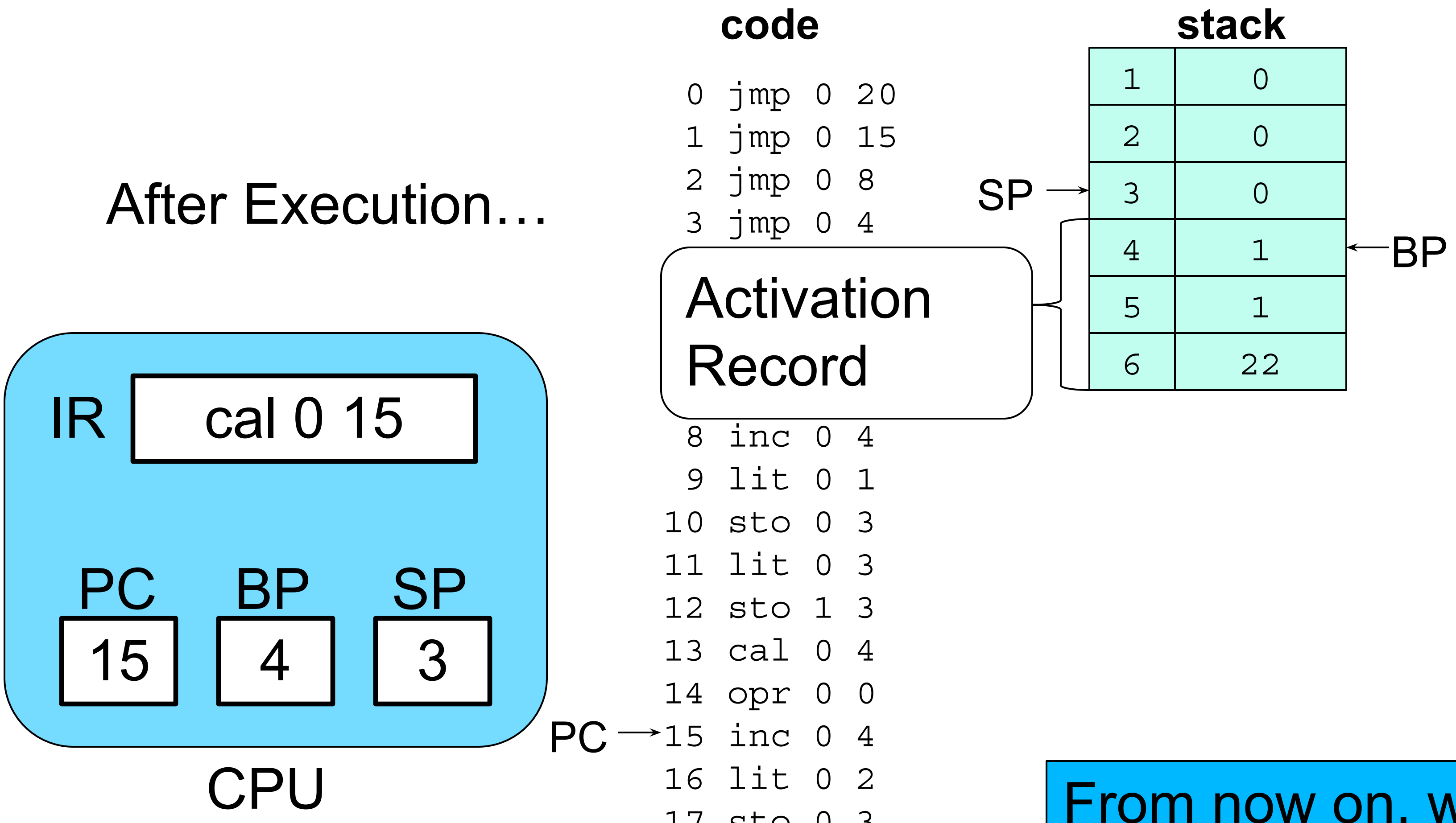
stack

	1	0	
	2	0	
SP →	3	0	
	4	1	← BP
	5	1	
	6	22	

CAL L, M →

```
stack[sp + 1] ← base(L);
     stack[sp + 2] ← bp;
     stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

After Execution…



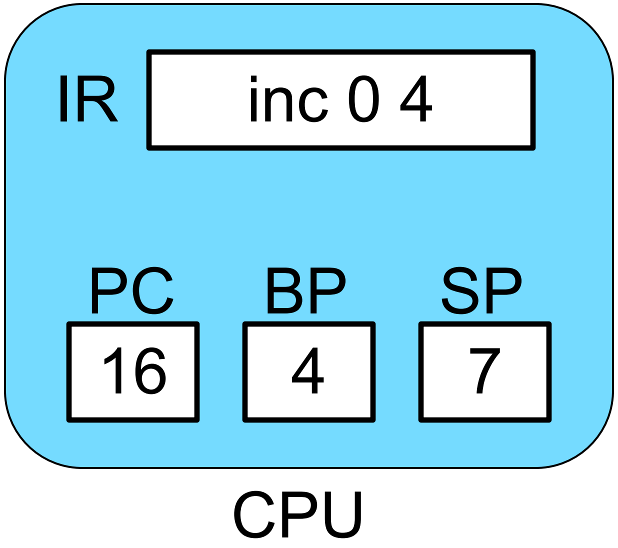
code

```
 0 jmp 0 20
 1 jmp 0 15
 2 jmp 0 8
 3 jmp 0 4
 8 inc 0 4
 9 lit 0 1
10 sto 0 3
11 lit 0 3
12 sto 1 3
13 cal 0 4
14 opr 0 0
15 inc 0 4
16 lit 0 2
17 sto 0 3
18 cal 0 8
19 opr 0 0
20 inc 0 3
21 cal 0 15
22 opr 0 0
```

stack

1	0
2	0
3	0
4	1
5	1
6	22

From now on, we'll only show the result after the instruction have been executed.



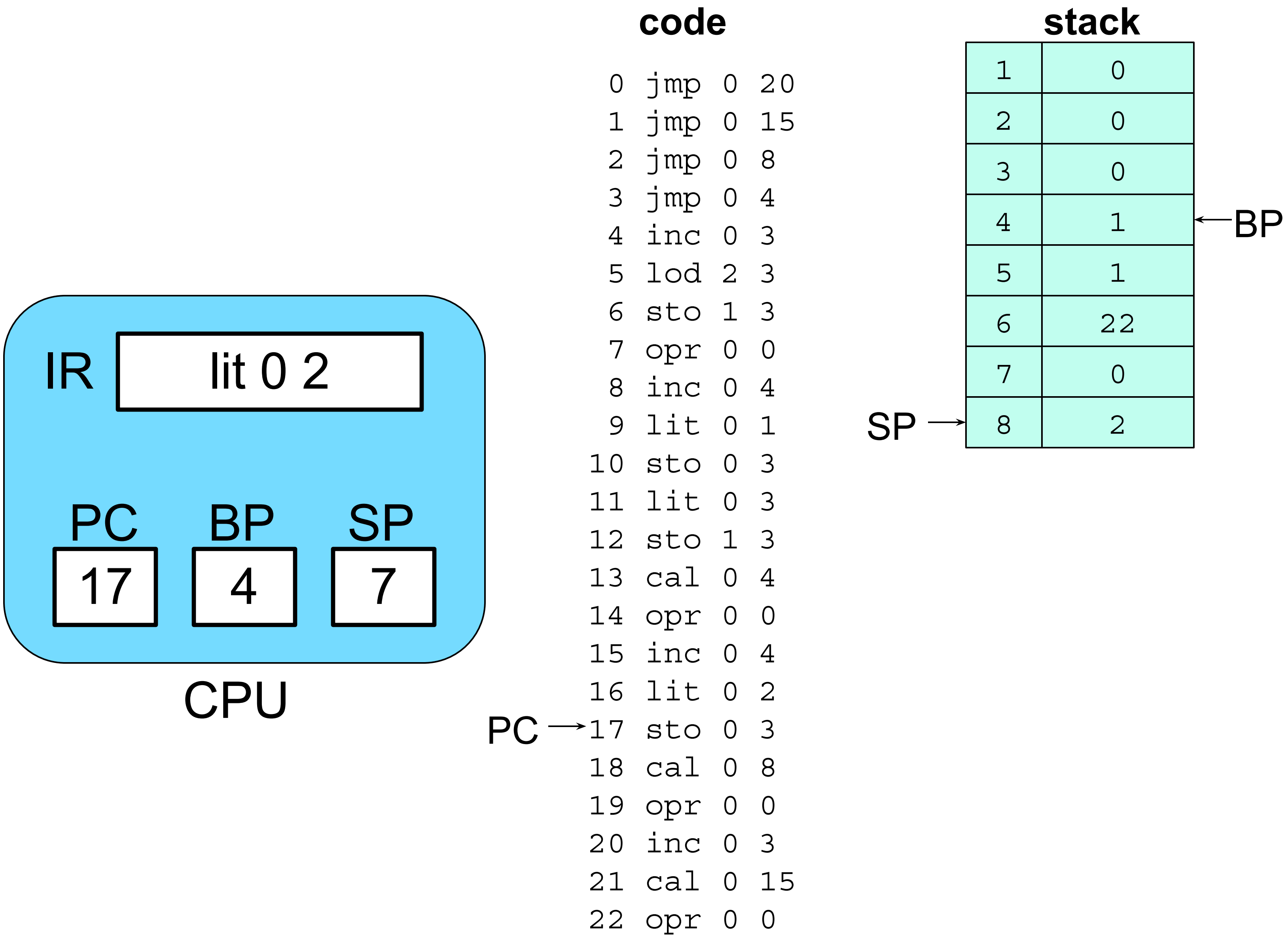
code

```
       0  jmp  0  20
       1  jmp  0  15
       2  jmp  0  8
       3  jmp  0  4
       4  inc  0  3
       5  lod  2  3
       6  sto  1  3
       7  opr  0  0
       8  inc  0  4
       9  lit  0  1
      10  sto  0  3
      11  lit  0  3
      12  sto  1  3
      13  cal  0  4
      14  opr  0  0
      15  inc  0  4
PC → 16  lit  0  2
      17  sto  0  3
      18  cal  0  8
      19  opr  0  0
      20  inc  0  3
      21  cal  0  15
      22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	← BP
	5	1	
	6	22	
SP →	7	0	

We are “reserving space” for activation record and one variable.

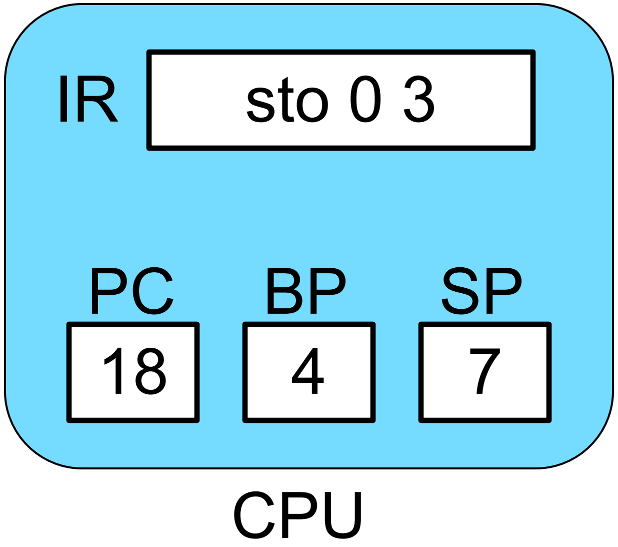


code

```
 0 jmp 0 20
 1 jmp 0 15
 2 jmp 0 8
 3 jmp 0 4
 4 inc 0 3
 5 lod 2 3
 6 sto 1 3
 7 opr 0 0
 8 inc 0 4
 9 lit 0 1
10 sto 0 3
11 lit 0 3
12 sto 1 3
13 cal 0 4
14 opr 0 0
15 inc 0 4
16 lit 0 2
17 sto 0 3
18 cal 0 8
19 opr 0 0
20 inc 0 3
21 cal 0 15
22 opr 0 0
```

stack

1	0
2	0
3	0
4	1
5	1
6	22
7	0
8	2



IR: sto 0 3

PC	BP	SP
18	4	7

CPU

code

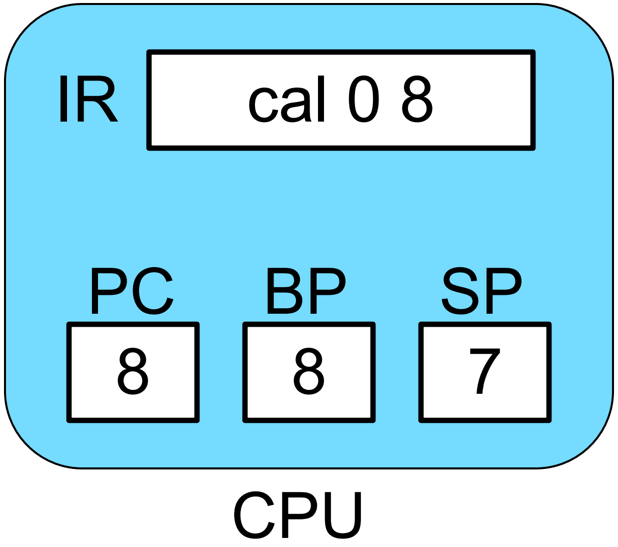
```
       0  jmp  0  20
       1  jmp  0  15
       2  jmp  0  8
       3  jmp  0  4
       4  inc  0  3
       5  lod  2  3
       6  sto  1  3
       7  opr  0  0
       8  inc  0  4
       9  lit  0  1
      10  sto  0  3
      11  lit  0  3
      12  sto  1  3
      13  cal  0  4
      14  opr  0  0
      15  inc  0  4
      16  lit  0  2
      17  sto  0  3
PC → 18  cal  0  8
      19  opr  0  0
      20  inc  0  3
      21  cal  0  15
      22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	← BP
	5	1	
	6	22	
SP →	7	2	
	8	2	

STO L, M →

stack[base(L,BP)+M] ← stack[sp];
    sp ← sp – 1;



code

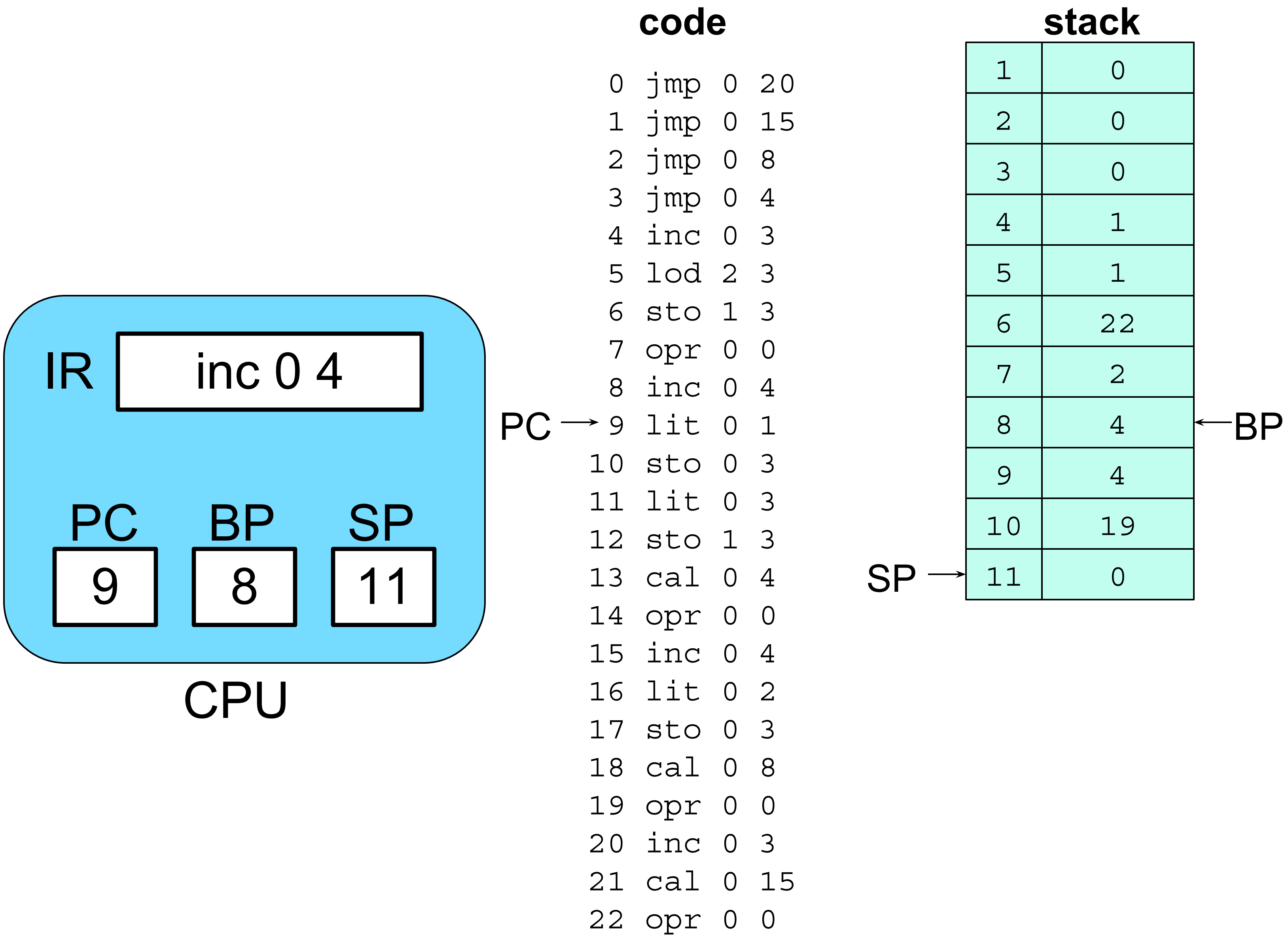
```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
PC →  8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
SP →	7	2	
	8	4	← BP
	9	4	
	10	19	

CAL L, M →

```
stack[sp + 1] ← base(L);
     stack[sp + 2] ← bp;
     stack[sp + 3] ← pc
bp ← sp + 1;
pc ← M;
```

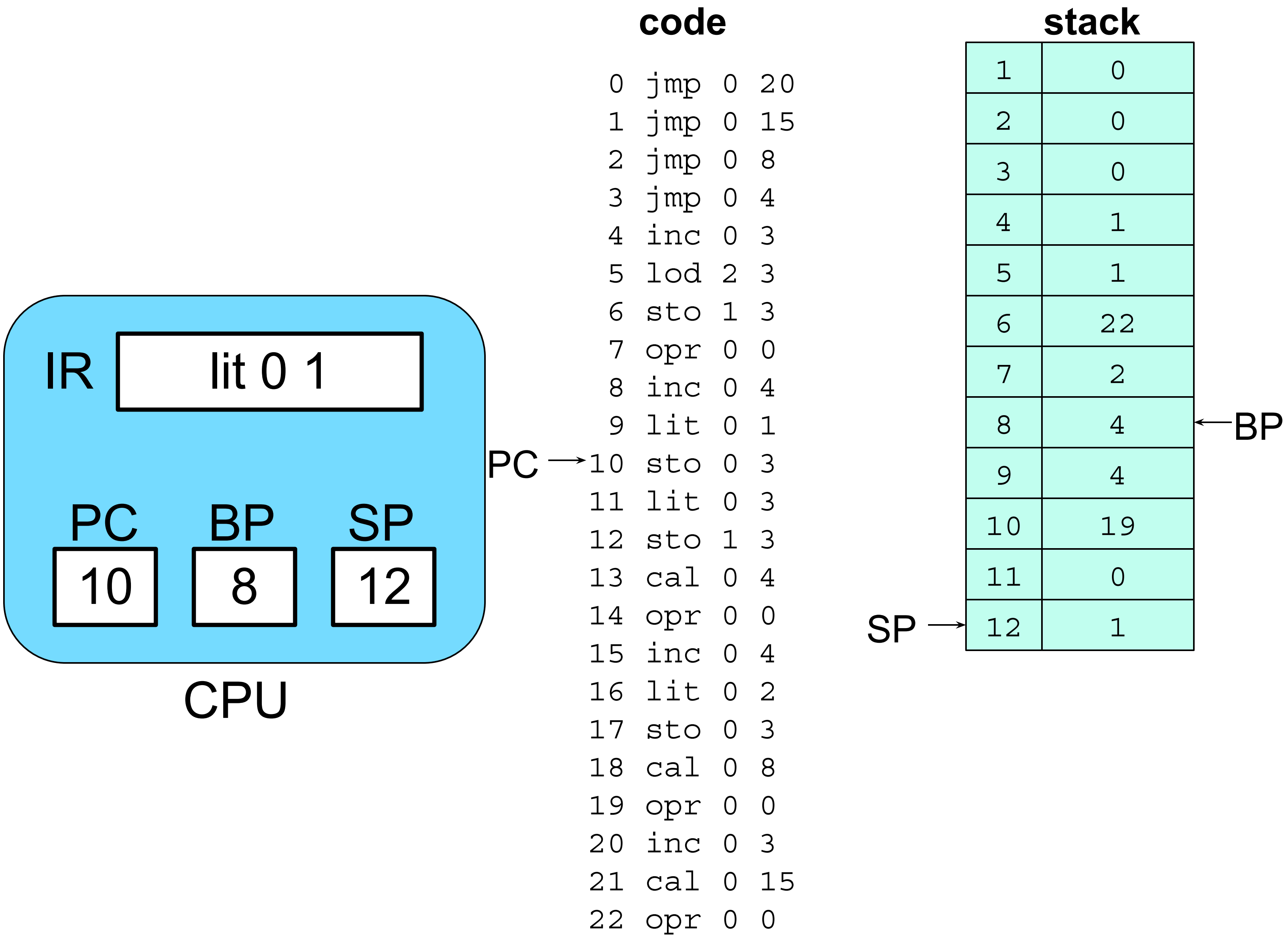


code

```
 0  jmp  0  20
 1  jmp  0  15
 2  jmp  0  8
 3  jmp  0  4
 4  inc  0  3
 5  lod  2  3
 6  sto  1  3
 7  opr  0  0
 8  inc  0  4
 9  lit  0  1
10  sto  0  3
11  lit  0  3
12  sto  1  3
13  cal  0  4
14  opr  0  0
15  inc  0  4
16  lit  0  2
17  sto  0  3
18  cal  0  8
19  opr  0  0
20  inc  0  3
21  cal  0  15
22  opr  0  0
```

stack

1	0
2	0
3	0
4	1
5	1
6	22
7	2
8	4
9	4
10	19
11	0

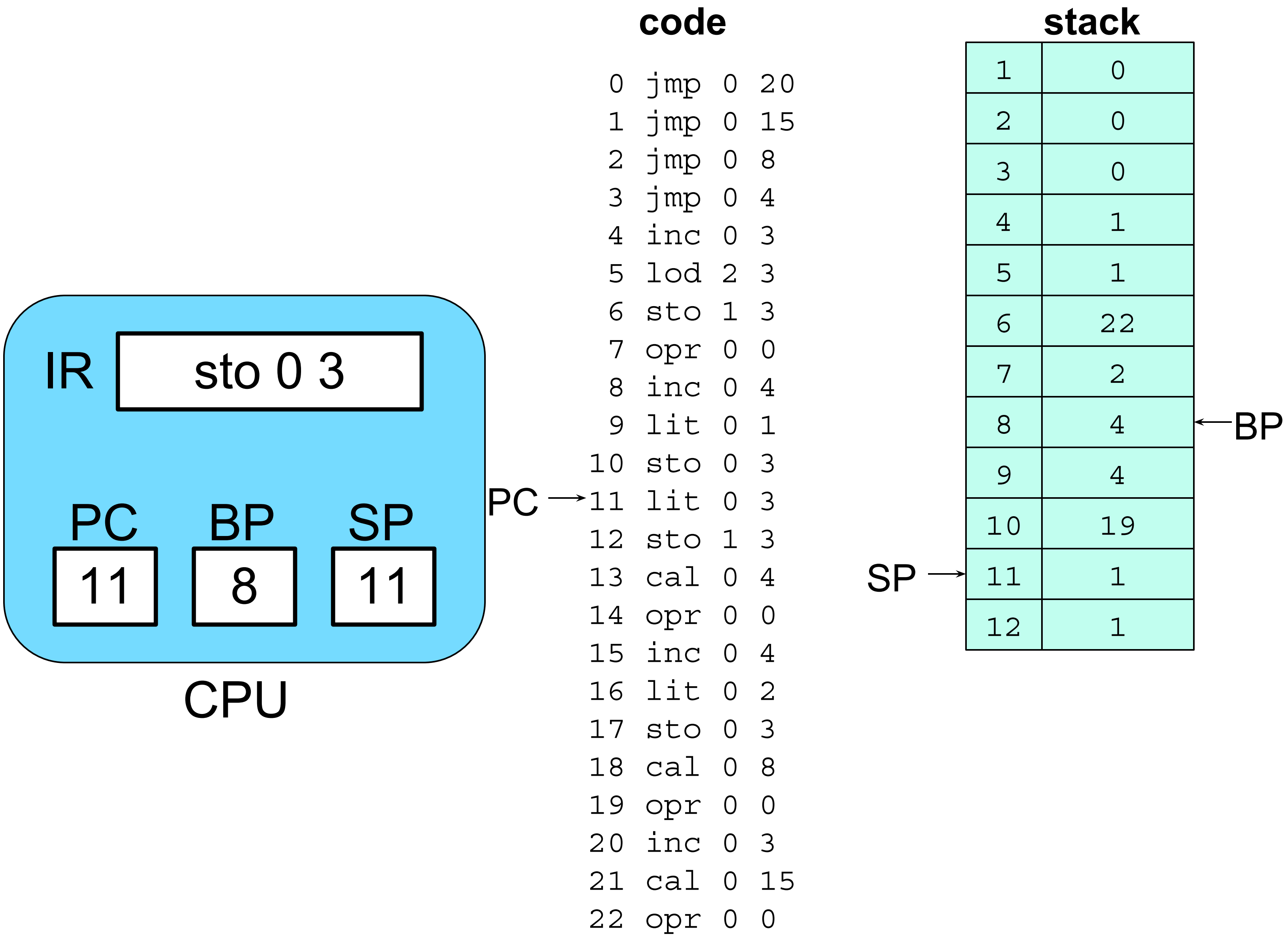


code

```
     0  jmp  0  20
     1  jmp  0  15
     2  jmp  0  8
     3  jmp  0  4
     4  inc  0  3
     5  lod  2  3
     6  sto  1  3
     7  opr  0  0
     8  inc  0  4
     9  lit  0  1
PC →10  sto  0  3
    11  lit  0  3
    12  sto  1  3
    13  cal  0  4
    14  opr  0  0
    15  inc  0  4
    16  lit  0  2
    17  sto  0  3
    18  cal  0  8
    19  opr  0  0
    20  inc  0  3
    21  cal  0  15
    22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	2	
	8	4	← BP
	9	4	
	10	19	
	11	0	
SP →	12	1	



IR: sto 0 3

PC	BP	SP
11	8	11

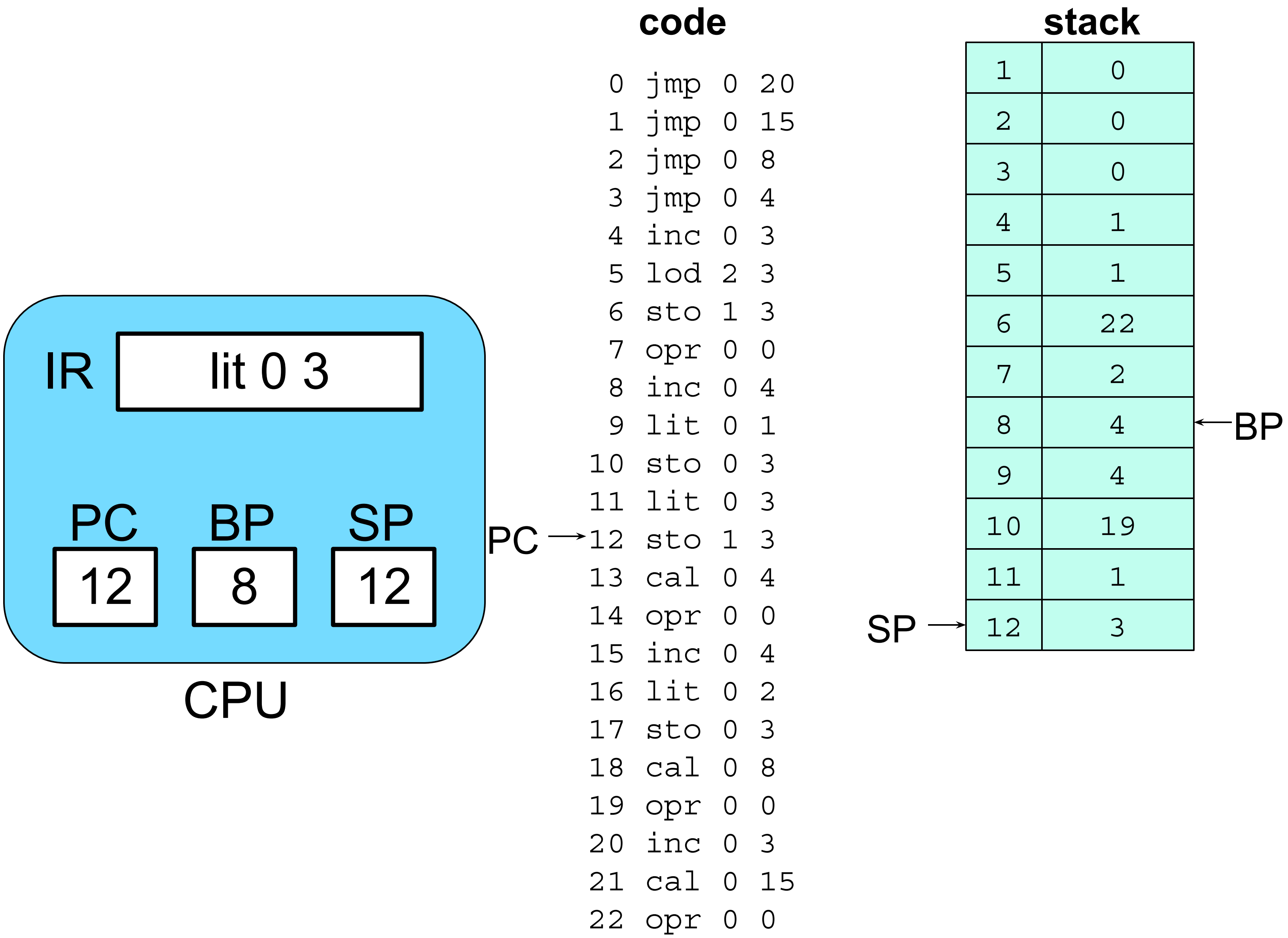
CPU

code

```
    0  jmp  0  20
    1  jmp  0  15
    2  jmp  0  8
    3  jmp  0  4
    4  inc  0  3
    5  lod  2  3
    6  sto  1  3
    7  opr  0  0
    8  inc  0  4
    9  lit  0  1
   10  sto  0  3
PC →11  lit  0  3
   12  sto  1  3
   13  cal  0  4
   14  opr  0  0
   15  inc  0  4
   16  lit  0  2
   17  sto  0  3
   18  cal  0  8
   19  opr  0  0
   20  inc  0  3
   21  cal  0  15
   22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	2	
	8	4	← BP
	9	4	
	10	19	
SP →	11	1	
	12	1	



CPU

IR: lit 0 3

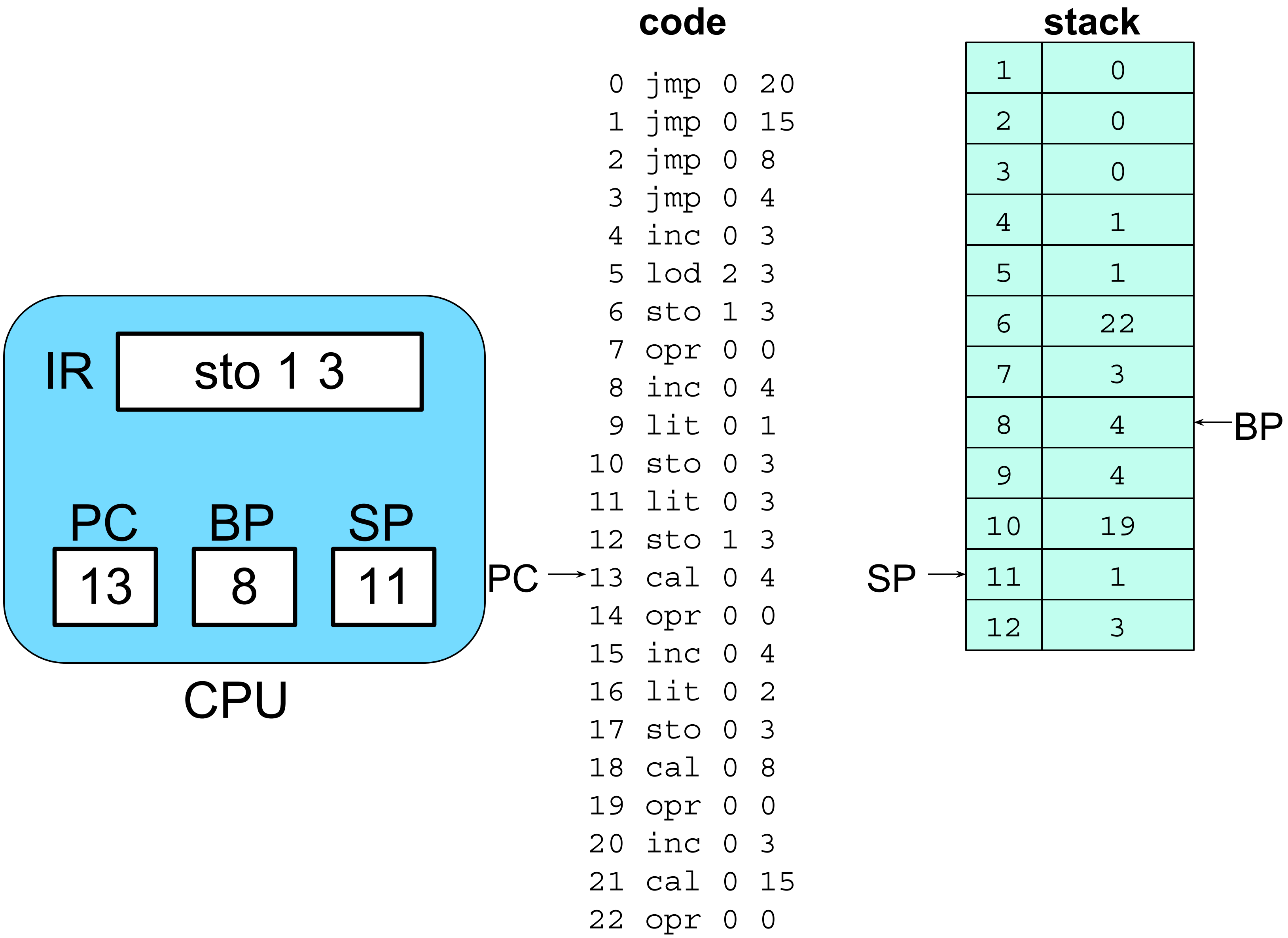
PC	BP	SP
12	8	12

code

```
 0  jmp  0  20
 1  jmp  0  15
 2  jmp  0  8
 3  jmp  0  4
 4  inc  0  3
 5  lod  2  3
 6  sto  1  3
 7  opr  0  0
 8  inc  0  4
 9  lit  0  1
10  sto  0  3
11  lit  0  3
12  sto  1  3
13  cal  0  4
14  opr  0  0
15  inc  0  4
16  lit  0  2
17  sto  0  3
18  cal  0  8
19  opr  0  0
20  inc  0  3
21  cal  0  15
22  opr  0  0
```

stack

1	0
2	0
3	0
4	1
5	1
6	22
7	2
8	4
9	4
10	19
11	1
12	3



IR: sto 1 3

PC: 13 | BP: 8 | SP: 11

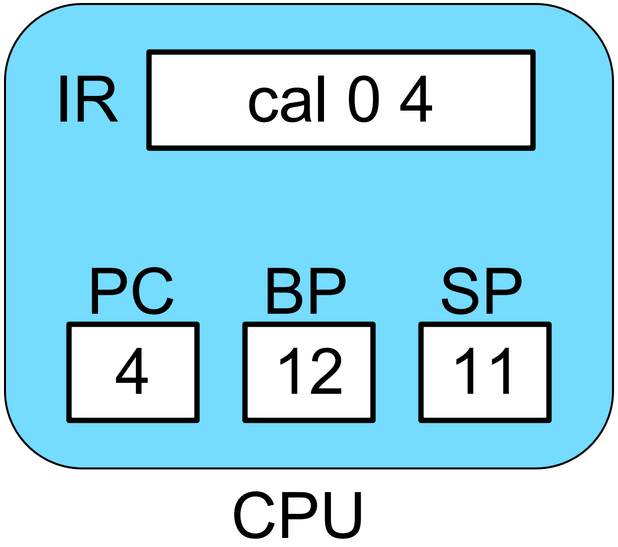
CPU

code

```
    0  jmp  0  20
    1  jmp  0  15
    2  jmp  0  8
    3  jmp  0  4
    4  inc  0  3
    5  lod  2  3
    6  sto  1  3
    7  opr  0  0
    8  inc  0  4
    9  lit  0  1
   10  sto  0  3
   11  lit  0  3
   12  sto  1  3
PC →13  cal  0  4
   14  opr  0  0
   15  inc  0  4
   16  lit  0  2
   17  sto  0  3
   18  cal  0  8
   19  opr  0  0
   20  inc  0  3
   21  cal  0  15
   22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	← BP
	9	4	
	10	19	
SP →	11	1	
	12	3	

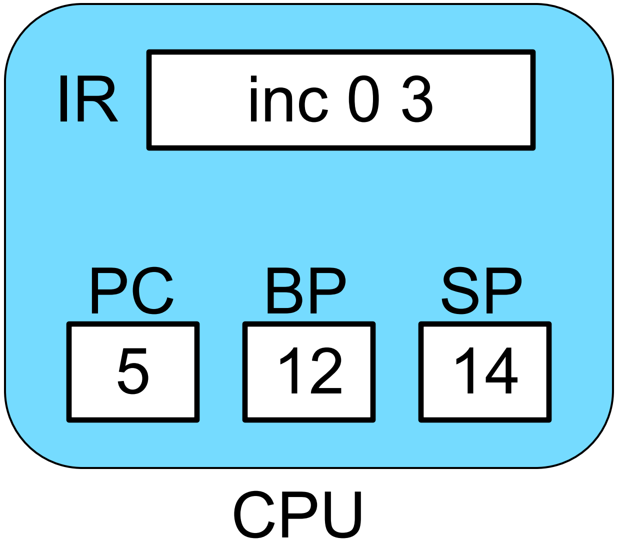


code

```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
PC → 4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	
	9	4	
	10	19	
SP →	11	1	
	12	8	← BP
	13	8	
	14	14	



IR inc 0 3

PC 5

BP 12

SP 14

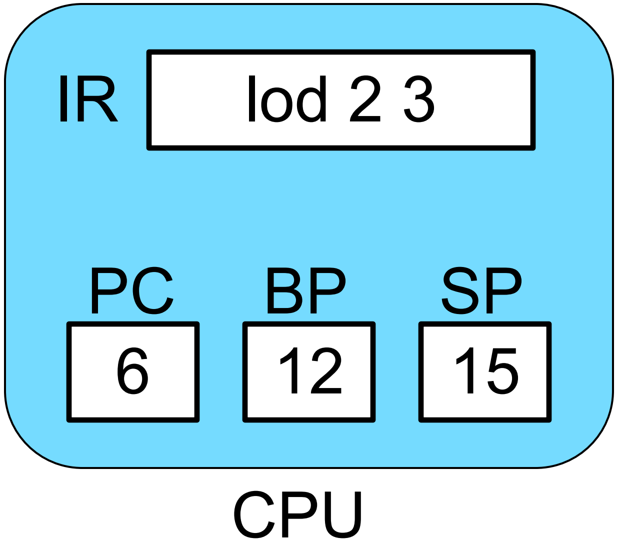
CPU

code

```
   0  jmp  0  20
   1  jmp  0  15
   2  jmp  0  8
   3  jmp  0  4
   4  inc  0  3
PC → 5  lod  2  3
   6  sto  1  3
   7  opr  0  0
   8  inc  0  4
   9  lit  0  1
  10  sto  0  3
  11  lit  0  3
  12  sto  1  3
  13  cal  0  4
  14  opr  0  0
  15  inc  0  4
  16  lit  0  2
  17  sto  0  3
  18  cal  0  8
  19  opr  0  0
  20  inc  0  3
  21  cal  0  15
  22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	
	9	4	
	10	19	
	11	1	
	12	8	← BP
	13	8	
SP →	14	14	

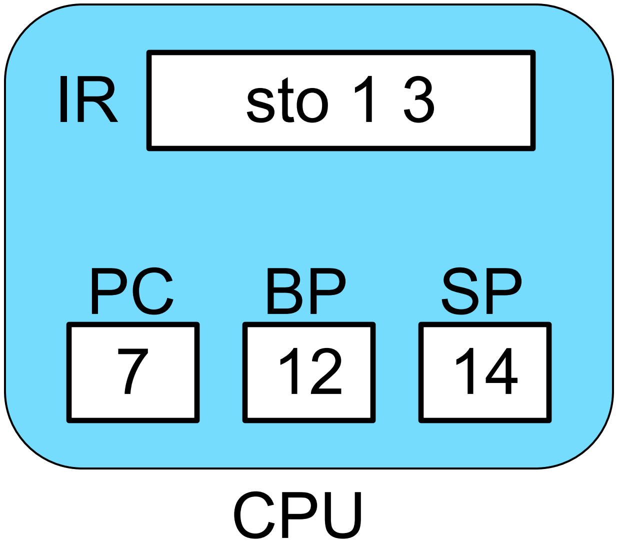


code

```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
PC →  6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	
	9	4	
	10	19	
	11	1	
	12	8	← BP
	13	8	
	14	14	
SP →	15	3	



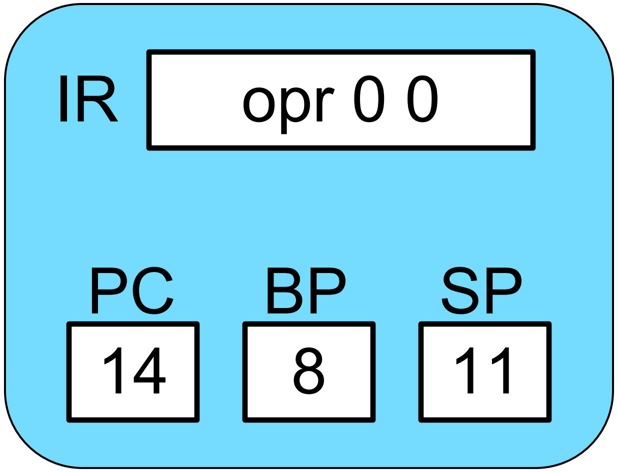
code

```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
PC →  7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
     19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	
	9	4	
	10	19	
	11	3	
	12	8	← BP
	13	8	
SP →	14	14	
	15	3	

RTN 0,0 →
sp ← bp-1;
pc ← stack[sp+3];
bp ← stack[sp+2];



CPU

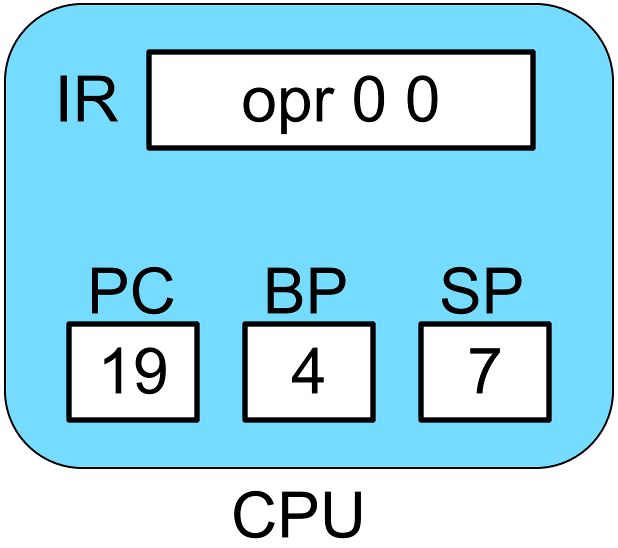
code

```
 0  jmp  0  20
 1  jmp  0  15
 2  jmp  0  8
 3  jmp  0  4
 4  inc  0  3
 5  lod  2  3
 6  sto  1  3
 7  opr  0  0
 8  inc  0  4
 9  lit  0  1
10  sto  0  3
11  lit  0  3
12  sto  1  3
13  cal  0  4
PC →14  opr  0  0
15  inc  0  4
16  lit  0  2
17  sto  0  3
18  cal  0  8
19  opr  0  0
20  inc  0  3
21  cal  0  15
22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	←BP
	9	4	
	10	19	
SP →	11	3	
	12	8	
	13	8	
	14	14	
	15	3	

RTN 0,0 →
sp ← bp-1;
pc ← stack[sp+3];
bp ← stack[sp+2];



IR: opr 0 0

PC	BP	SP
19	4	7

CPU

RTN 0,0 →

sp ← bp-1;

pc ← stack[sp+3];

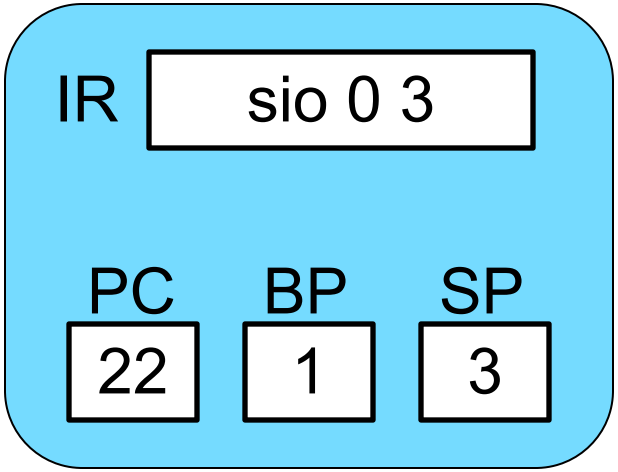
bp ← stack[sp+2];

code

```
      0  jmp  0  20
      1  jmp  0  15
      2  jmp  0  8
      3  jmp  0  4
      4  inc  0  3
      5  lod  2  3
      6  sto  1  3
      7  opr  0  0
      8  inc  0  4
      9  lit  0  1
     10  sto  0  3
     11  lit  0  3
     12  sto  1  3
     13  cal  0  4
     14  opr  0  0
     15  inc  0  4
     16  lit  0  2
     17  sto  0  3
     18  cal  0  8
PC → 19  opr  0  0
     20  inc  0  3
     21  cal  0  15
     22  opr  0  0
```

stack

	1	0	
	2	0	
	3	0	
	4	1	← BP
	5	1	
	6	22	
SP →	7	3	
	8	4	
	9	4	
	10	19	
	11	3	
	12	8	
	13	8	
	14	14	
	15	3	



CPU

code

```
 0 jmp 0 20
 1 jmp 0 15
 2 jmp 0 8
 3 jmp 0 4
 4 inc 0 3
 5 lod 2 3
 6 sto 1 3
 7 opr 0 0
 8 inc 0 4
 9 lit 0 1
10 sto 0 3
11 lit 0 3
12 sto 1 3
13 cal 0 4
14 opr 0 0
15 inc 0 4
16 lit 0 2
17 sto 0 3
18 cal 0 8
19 opr 0 0
20 inc 0 3
21 cal 0 15
PC →22 opr 0 0
```

stack

	1	0	← BP
	2	0	
SP →	3	0	
	4	1	
	5	1	
	6	22	
	7	3	
	8	4	
	9	4	
	10	19	
	11	3	
	12	8	
	13	8	
	14	14	
	15	3	

RTN 0,0 →

sp ← bp-1;

pc ← stack[sp+3];

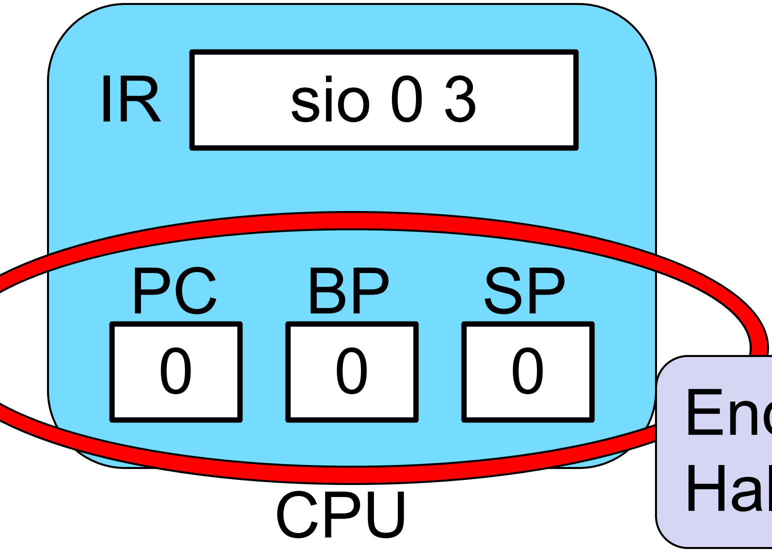
bp ← stack[sp+2];

code

```
PC → 0  jmp  0  20
     1  jmp  0  15
     2  jmp  0  8
     3  jmp  0  4
     4  inc  0  3
     5  lod  2  3
     6  sto  1  3
     7  opr  0  0
     8  inc  0  4
     9  lit  0  1
    10  sto  0  3
    11  lit  0  3
    12  sto  1  3
    13  cal  0  4
    17  sto  0  3
    18  cal  0  8
    19  opr  0  0
    20  inc  0  3
    21  cal  0  15
    22  opr  0  0
```

SP → **stack** ← BP

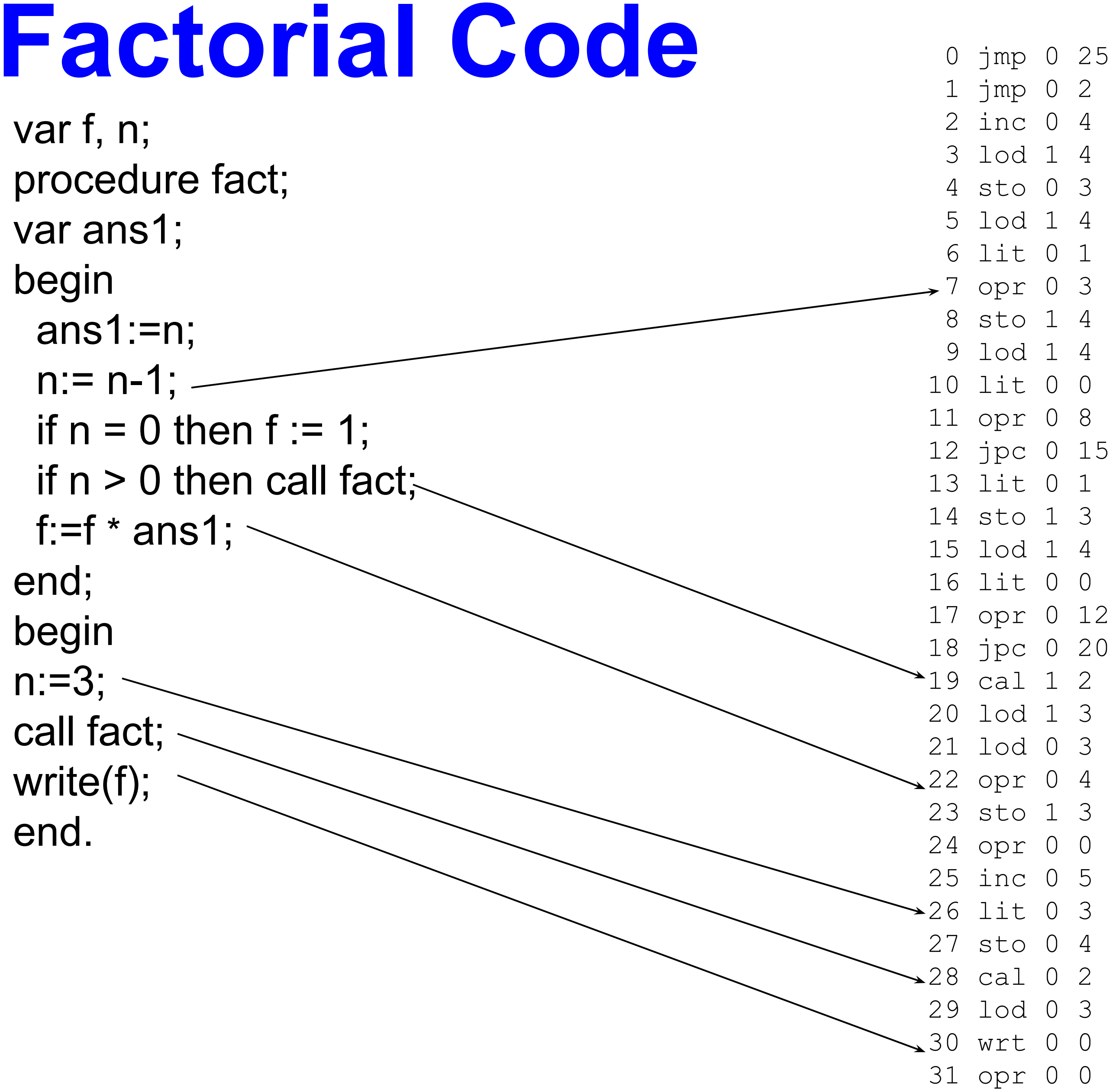
1	0
2	0
3	0
4	1
5	1
6	22
7	3
8	4
9	4
10	19
11	3
12	8
13	8
14	14
15	3



End State.
Halt P-Machine

RTN 0,0 →
sp ← bp-1;
pc ← stack[sp+3];
bp ← stack[sp+2];

Factorial Code



```
var f, n;
procedure fact;
var ans1;
begin
  ans1:=n;
  n:= n-1;
  if n = 0 then f := 1;
  if n > 0 then call fact;
  f:=f * ans1;
end;
begin
n:=3;
call fact;
write(f);
end.
```

```
 0 jmp 0 25
 1 jmp 0 2
 2 inc 0 4
 3 lod 1 4
 4 sto 0 3
 5 lod 1 4
 6 lit 0 1
 7 opr 0 3
 8 sto 1 4
 9 lod 1 4
10 lit 0 0
11 opr 0 8
12 jpc 0 15
13 lit 0 1
14 sto 1 3
15 lod 1 4
16 lit 0 0
17 opr 0 12
18 jpc 0 20
19 cal 1 2
20 lod 1 3
21 lod 0 3
22 opr 0 4
23 sto 1 3
24 opr 0 0
25 inc 0 5
26 lit 0 3
27 sto 0 4
28 cal 0 2
29 lod 0 3
30 wrt 0 0
31 opr 0 0
```

Running Factorial on PM/0

	pc	bp	sp	stack
Initial values	0	1	0	0 0 0
0 jmp 0, 25	25	1	0	0 0 0
25 inc 0, 5	26	1	5	0 0 0 0 0
26 lit 0, 3	27	1	6	0 0 0 0 0 3
27 sto 0, 4	28	1	5	0 0 0 0 3
28 cal 0, 2	2	6	5	0 0 0 0 3 | 1 1 29
2 inc 0, 4	3	6	9	0 0 0 0 3 | 1 1 29 0
3 lod 1, 4	4	6	10	0 0 0 0 3 | 1 1 29 0 3
4 sto 0, 3	5	6	9	0 0 0 0 3 | 1 1 29 3
5 lod 1, 4	6	6	10	0 0 0 0 3 | 1 1 29 3 3
6 lit 0, 1	7	6	11	0 0 0 0 3 | 1 1 29 3 3 1
7 opr 0, 3	8	6	10	0 0 0 0 3 | 1 1 29 3 2
8 sto 1, 4	9	6	9	0 0 0 0 2 | 1 1 29 3
9 lod 1, 4	10	6	10	0 0 0 0 2 | 1 1 29 3 2
10 lit 0, 0	11	6	11	0 0 0 0 2 | 1 1 29 3 2 0
11 opr 0, 8	12	6	10	0 0 0 0 2 | 1 1 29 3 0

code

```
0 jmp  0 25
1 jmp  0 2
2 inc  0 4
3 lod  1 4
4 sto  0 3
5 lod  1 4
6 lit  0 1
7 opr  0 3
8 sto  1 4
9 lod  1 4
10 lit 0 0
11 opr 0 8
12 jpc 0 15
13 lit 0 1
14 sto 1 3
15 lod 1 4
16 lit 0 0
```

Running a program on PM/0

		pc	bp	sp	stack
Initial values		0	1	0	0 0 0
12 jpc	0, 15	15	6	9	0 0 0 0 2| 1 1 29 3
15 lod	1, 4	16	6	10	0 0 0 0 2| 1 1 29 3 2
16 lit	0, 0	17	6	11	0 0 0 0 2| 1 1 29 3 2 0
17 opr	0, 12	18	6	10	0 0 0 0 2| 1 1 29 3 1
18 jpc	0, 20	19	6	9	0 0 0 0 2| 1 1 29 3
19 cal	1, 2	2	10	9	0 0 0 0 2| 1 1 29 3| 1 6 20
2 inc	0, 4	3	10	13	0 0 0 0 2| 1 1 29 3| 1 6 20 0
3 lod	2, 4	4	10	14	0 0 0 0 2| 1 1 29 3| 1 6 20 0 2
4 sto	0, 3	5	10	13	0 0 0 0 2| 1 1 29 3| 1 6 20 2
5 lod	2, 4	6	10	14	0 0 0 0 2| 1 1 29 3| 1 6 20 2 2
6 lit	0, 1	7	10	15	0 0 0 0 2| 1 1 29 3| 1 6 20 2 2 1
7 opr	0, 3	8	10	14	0 0 0 0 2| 1 1 29 3| 1 6 20 2 1
8 sto	2, 4	9	10	13	0 0 0 0 1| 1 1 29 3| 1 6 20 2
9 lod	2, 4	10	10	14	0 0 0 0 1| 1 1 29 3| 1 6 20 2 1
10 lit	0, 0	11	10	15	0 0 0 0 1| 1 1 29 3| 1 6 20 2 1 0

code

```
17 opr  0 12
18 jpc  0 20
19 cal 1 2
20 lod 1 3
21 lod 0 3
22 opr 0 4
23 sto 1 3
24 opr 0 0
25 inc 0 5
26 lit 0 3
27 sto 0 4
28 cal 0 2
29 lod 0 3
30 wrt 0 0
31 opr 0 0
```

Running a program on PM/0

		pc	bp	sp	stack
Initial values		0	1	0	0 0 0
11 opr	0, 8	12	10	14	0 0 0 0 1| 1 1 29 3| 1 6 20 2 0
12 jpc	0, 15	15	10	13	0 0 0 0 1| 1 1 29 3| 1 6 20 2
15 lod	2, 4	16	10	14	0 0 0 0 1| 1 1 29 3| 1 6 20 2 1
16 lit	0, 0	17	10	15	0 0 0 0 1| 1 1 29 3| 1 6 20 2 1 0
17 opr	0, 12	18	10	14	0 0 0 0 1| 1 1 29 3| 1 6 20 2 1
18 jpc	0, 20	19	10	13	0 0 0 0 1| 1 1 29 3| 1 6 20 2
19 cal	1, 2	2	14	13	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20
2 inc	0, 4	3	14	17	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 0
3 lod	3, 4	4	14	18	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 0 1
4 sto	0, 3	5	14	17	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 1
5 lod	3, 4	6	14	18	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 1 1
6 lit	0, 1	7	14	19	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 1 1 1
7 opr	0, 3	8	14	18	0 0 0 0 1| 1 1 29 3| 1 6 20 2| 1 10 20 1 0
8 sto	3, 4	9	14	17	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1
9 lod	3, 4	10	14	18	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 0

Running a program on PM/0

	pc	bp	sp	stack
Initial values	0	1	0	0 0 0
10 lit 0, 0	11	14	19	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 0 0
11 opr 0, 8	12	14	18	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 1
12 jpc 0, 15	13	14	17	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1
13 lit 0, 1	14	14	18	0 0 0 0 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 1
14 sto 3, 3	15	14	17	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1
15 lod 3, 4	16	14	18	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 0
16 lit 0, 0	17	14	19	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 0 0
17 opr 0, 12	18	14	18	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 0
18 jpc 0, 20	20	14	17	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1
20 lod 3, 3	21	14	18	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 1
21 lod 0, 3	22	14	19	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 1 1
22 opr 0, 4	23	14	18	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1 1
23 sto 3, 3	24	14	17	0 0 0 1 0| 1 1 29 3| 1 6 20 2| 1 10 20 1
24 opr 0, 0	20	10	13	0 0 0 1 0| 1 1 29 3| 1 6 20 2
20 lod 2, 3	21	10	14	0 0 0 1 0| 1 1 29 3| 1 6 20 2 1

Running a program on PM/0

	pc	bp	sp	stack
Initial values	0	1	0	0 0 0
21 lod 0, 3	22	10	15	0 0 0 1 0| 1 1 29 3| 1 6 20 2 1 2
22 opr 0, 4	23	10	14	0 0 0 1 0| 1 1 29 3| 1 6 20 2 2
23 sto 2, 3	24	10	13	0 0 0 2 0| 1 1 29 3| 1 6 20 2
24 opr 0, 0	20	6	9	0 0 0 2 0| 1 1 29 3
20 lod 1, 3	21	6	10	0 0 0 2 0| 1 1 29 3 2
21 lod 0, 3	22	6	11	0 0 0 2 0| 1 1 29 3 2 3
22 opr 0, 4	23	6	10	0 0 0 2 0| 1 1 29 3 6
23 sto 1, 3	24	6	9	0 0 0 6 0| 1 1 29 3
24 opr 0, 0	29	1	5	0 0 0 6 0
29 lod 0, 3	30	1	6	0 0 0 6 0 6
30 wrt 0, 0	31	1	5	0 0 0 6 0
31 opr 0, 0	0	0	0	