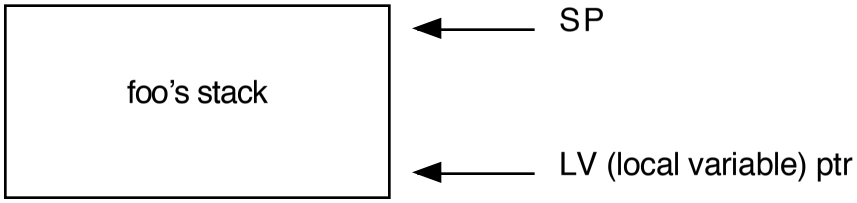
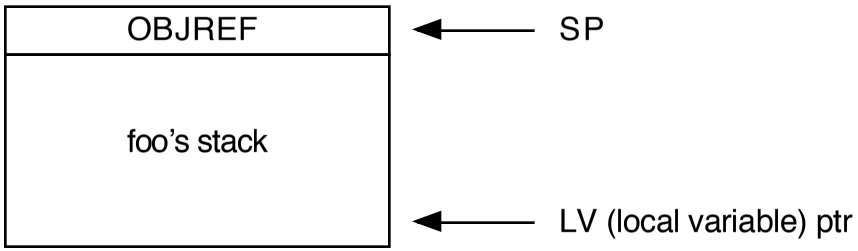


activity:

• presently executing method "foo". It wants to call method "bar".

INVOKEVIRTUAL STEP 1



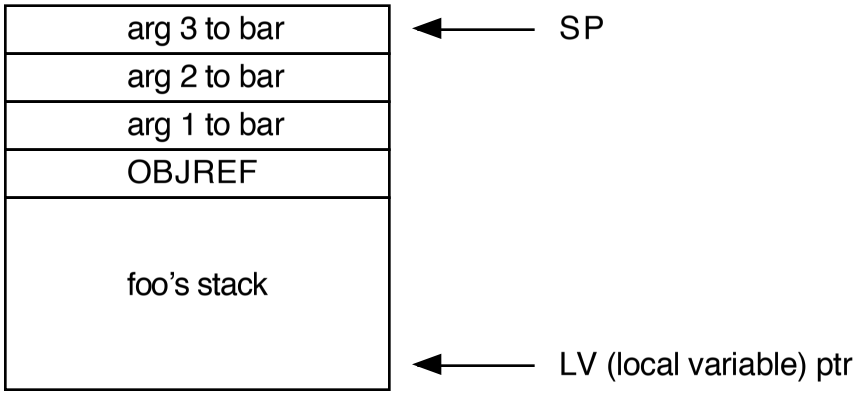
activity:

• method foo pushes OBJREF, which is a ptr to object from which it will call the bar method.

code:

```
BIPUSH <OBJREF>
```

INVOKEVIRTUAL STEP 2



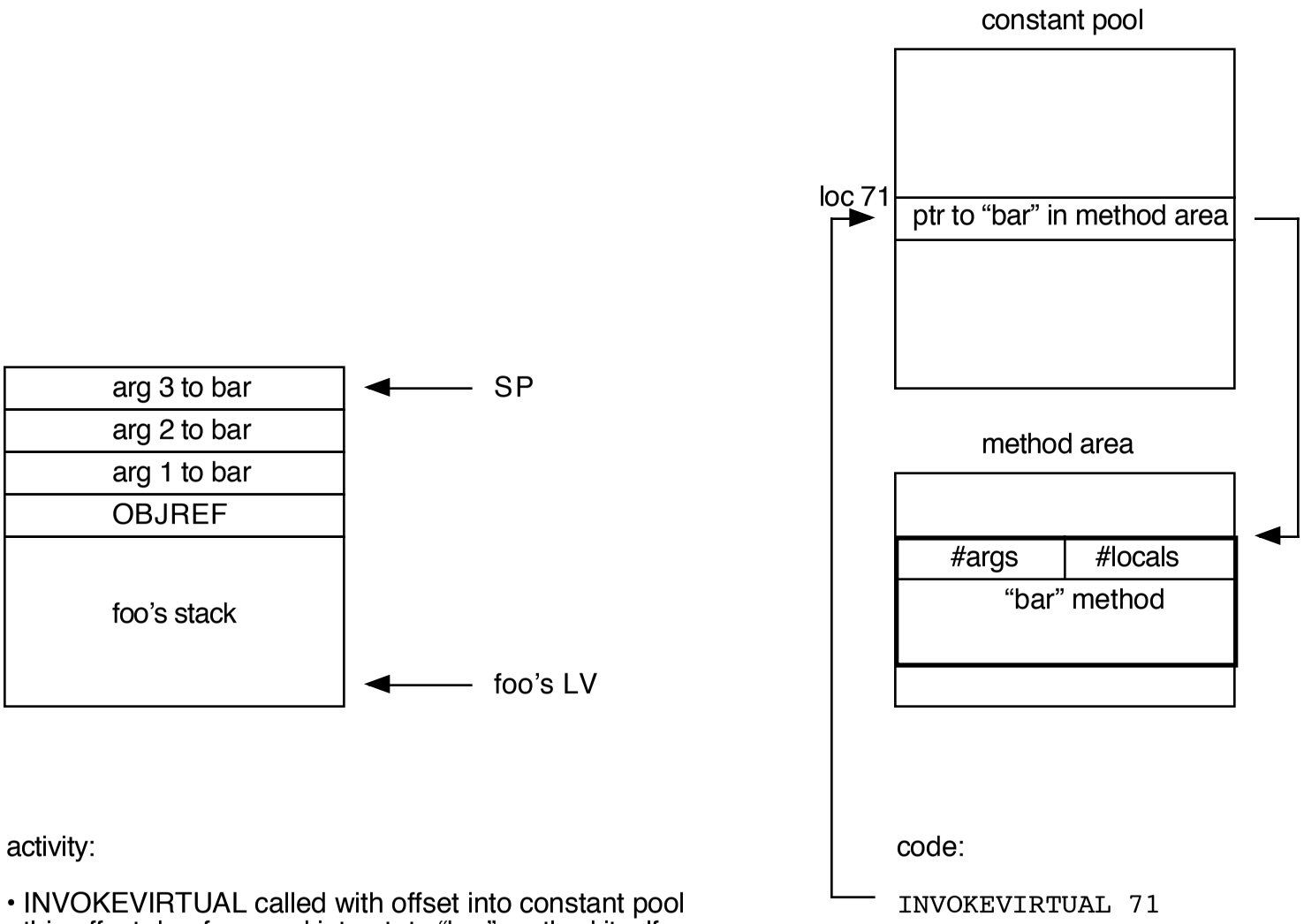
activity:

• method foo pushes three arguments to bar.

code:

```
BIPUSH <ARG1>
BIPUSH <ARG2>
BIPUSH <ARG3>
```

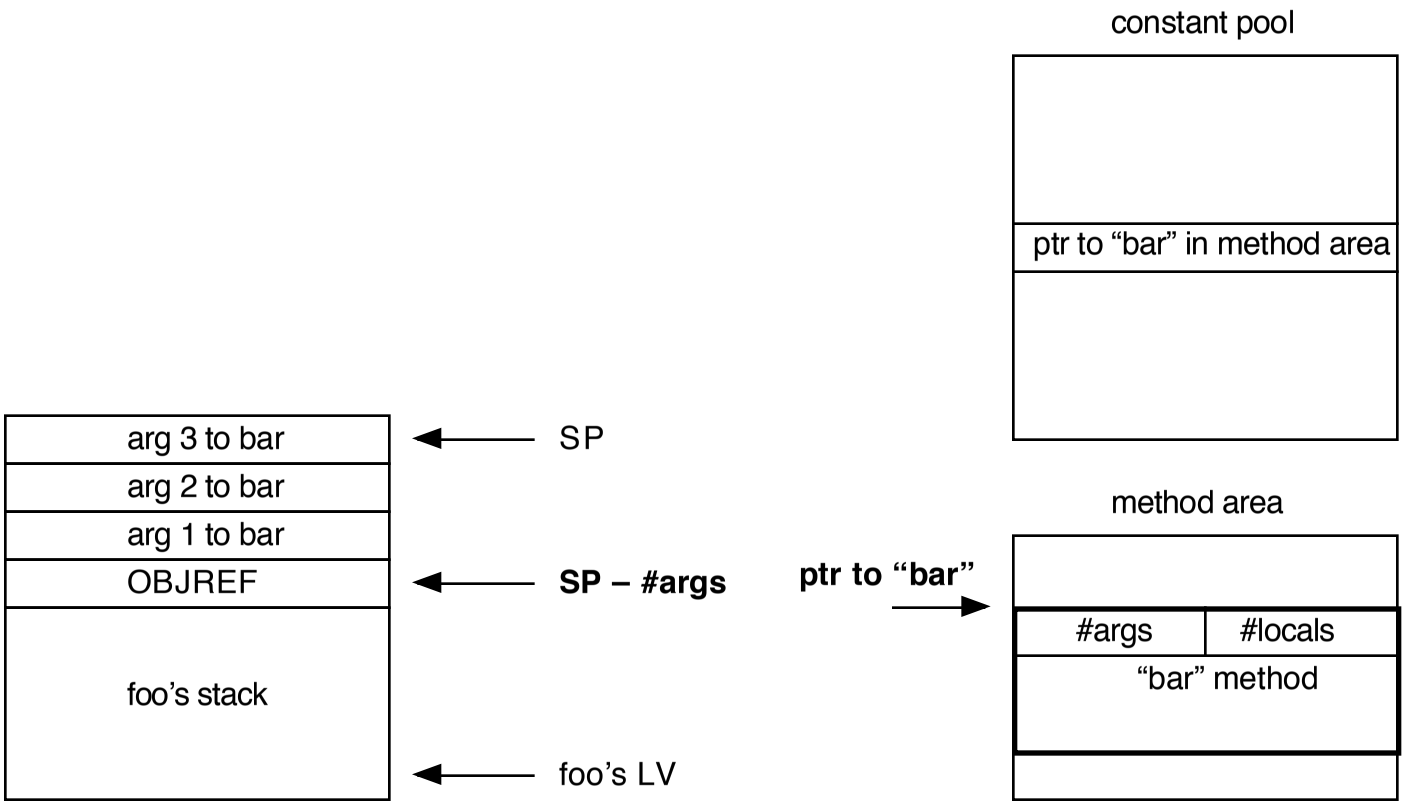
INVOKEVIRTUAL STEP 3



activity:

- INVOKEVIRTUAL called with offset into constant pool
- this offset dereferenced into ptr to “bar” method itself
- look into bar to determine how many args it accepts
- subtract this from SP to get ptr to base of arg stack for bar

INVOKEVIRTUAL STEP 4



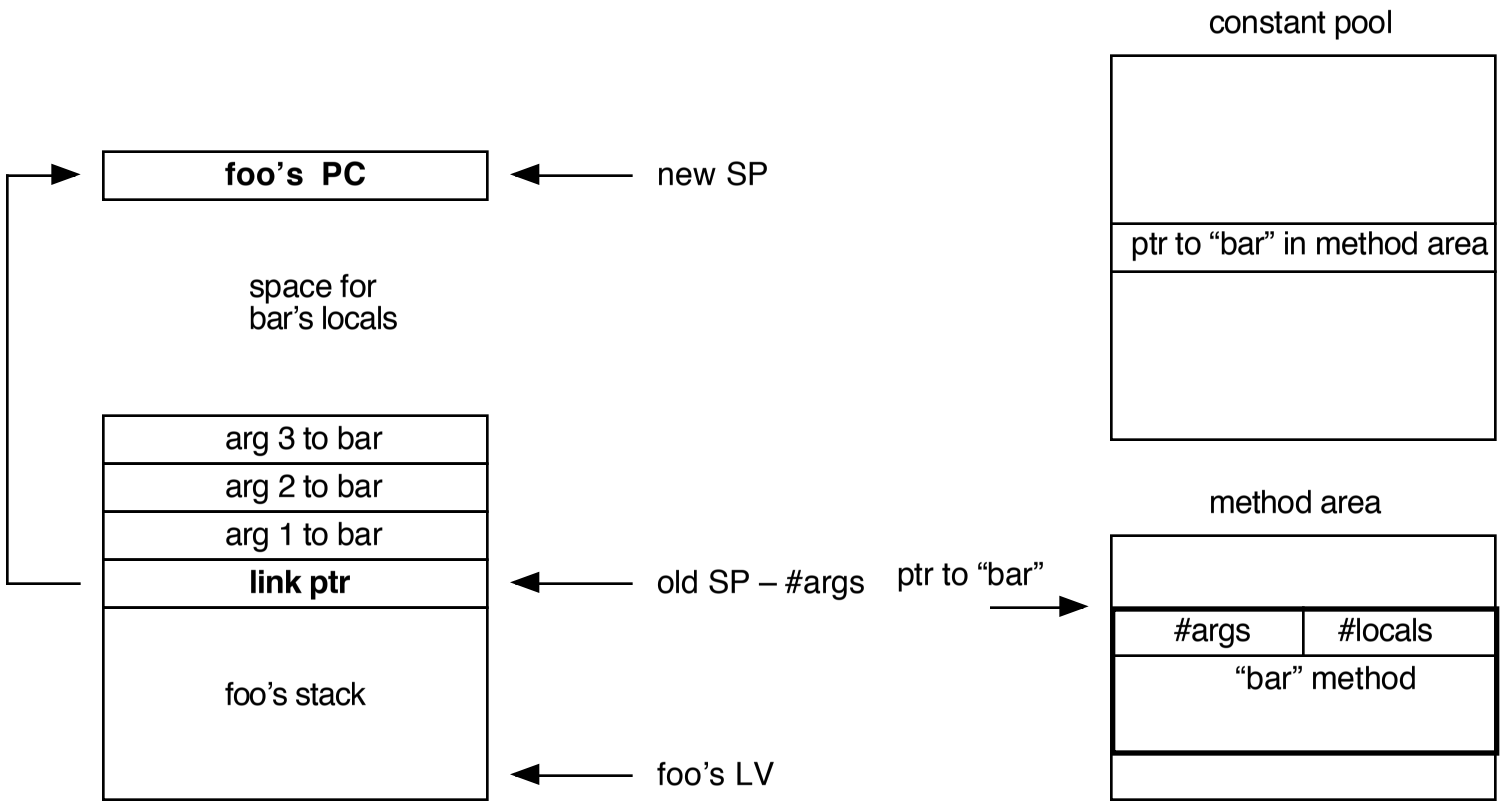
activity:

- have ptr to “bar”
- look into bar to determine how many args it accepts
- subtract this from SP to get ptr to base of arg stack for bar

code:

```
INVOKEVIRTUAL 71
```

INVOKEVIRTUAL STEP 5



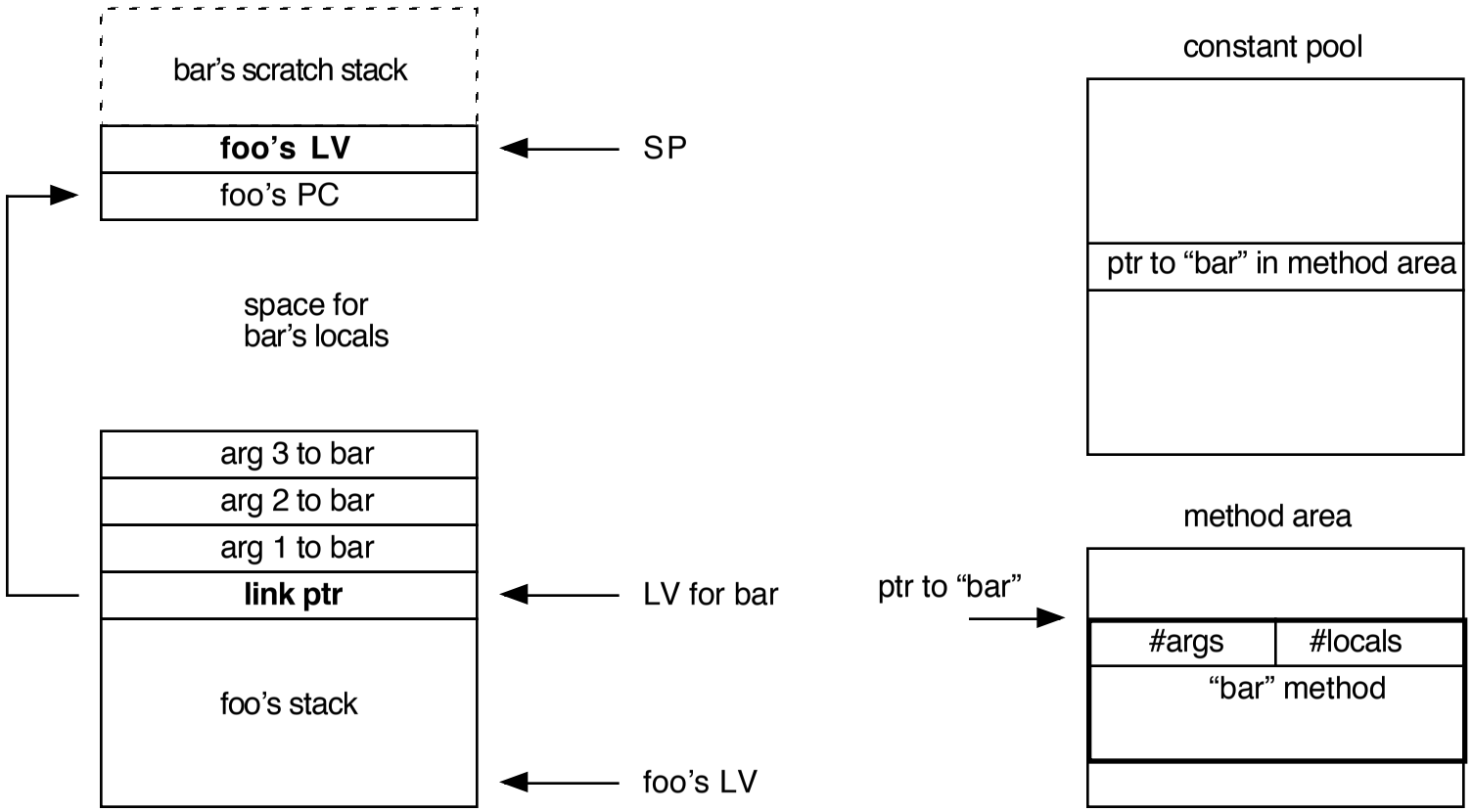
activity:

- have ptr (old SP – #args)
- add bar’s #args + #locals to reserve space for bar (new SP)
- where OBJREF lived, install link ptr
- save foo’s PC where link ptr is pointing

code:

```
INVOKEVIRTUAL 71
```

INVOKEVIRTUAL STEP 6



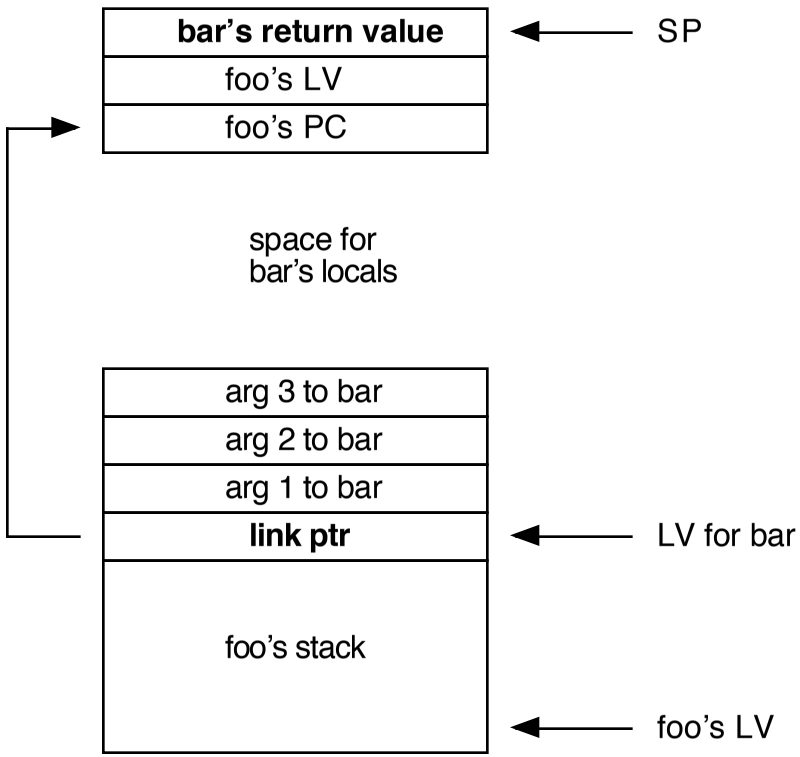
activity:

- push foo's LV on stack
- set new LV (for bar's context) at link ptr location
- begin executing bar's code!

code:

```
INVOKEVIRTUAL 71
```

INVOKEVIRTUAL STEP 7



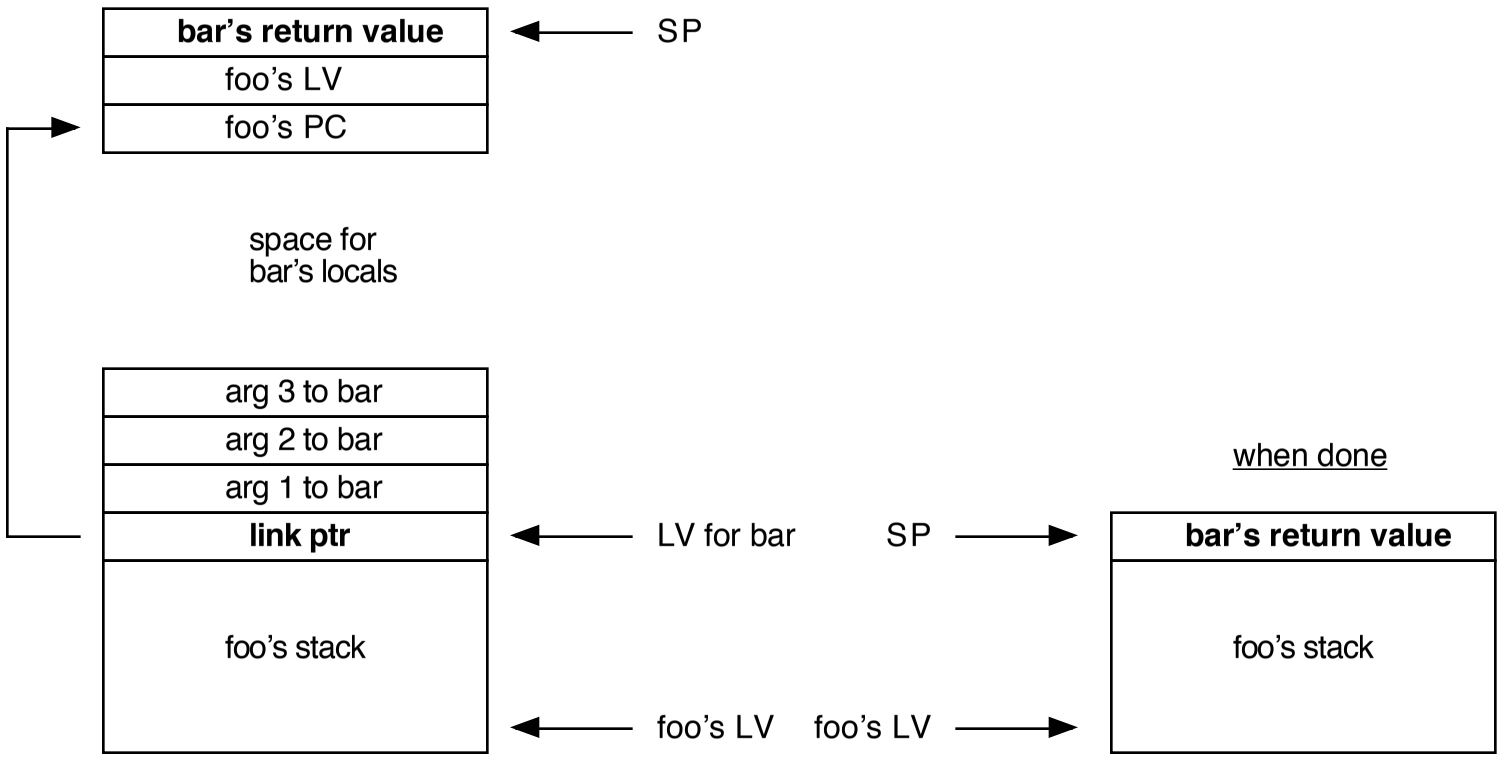
activity:

• bar has cleared its scratch stack and pushed its return value
• bar executes IRETURN

IJVM code:

```
IRETURN
```

RETURN STEP 1



activity:

- dereference bar's LV to get link ptr, that's new SP
- bar's return value is already in TOS register
- temporarily set LV to get at foo's PC
- incr MAR to point at foo's LV
- set PC from rd 2 steps back, read in foo's LV, fetch new opcode from foo
- from 1st step, SP points at loc of link ptr
- restore foo's LV from 2 steps back
- write return value onto new TOS, execute opcode from foo

microcode code:

```
MAR = SP = LV; read

LV = MAR = MDR; rd
MAR = LV + 1
PC = MDR; rd; fetch

MAR = SP
LV = MDR
MDR = TOS; wr; goto Main1
```

RETURN COMPLETE