Lecture 2:
Static Link Demystified

Lexical Scoping

Restrictions on caller and callee

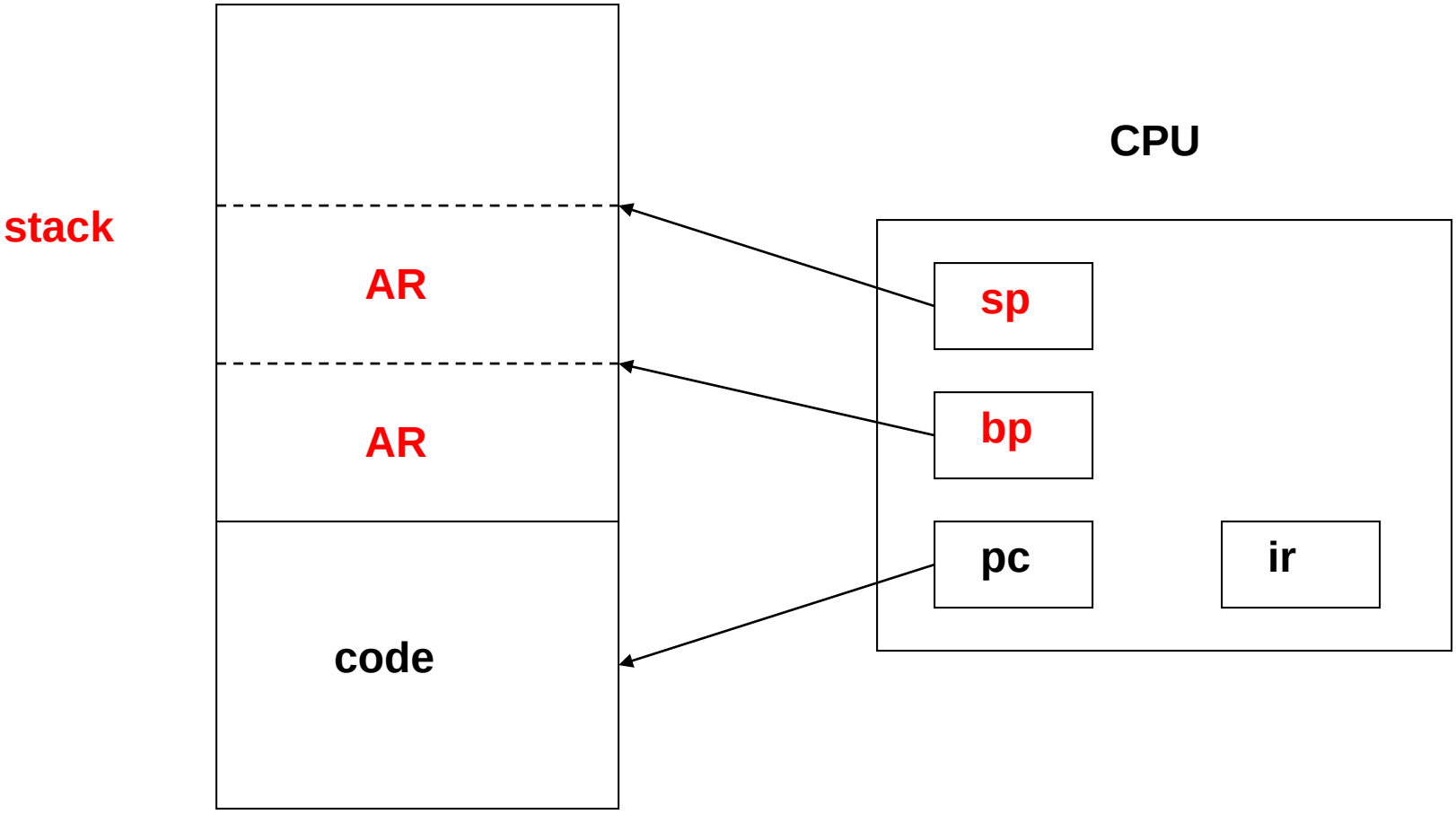
- a parent function can call a child function (but cannot call a grandchild function, cannot call a great-grand child function, etc.)
- a function can call itself recursively (or a sibling)
- a descendent function can call any ancestor function

Scoping

- a callee can only access the variables defined in its ancestor functions
- these variables are stored in ARs below the AR of the callee

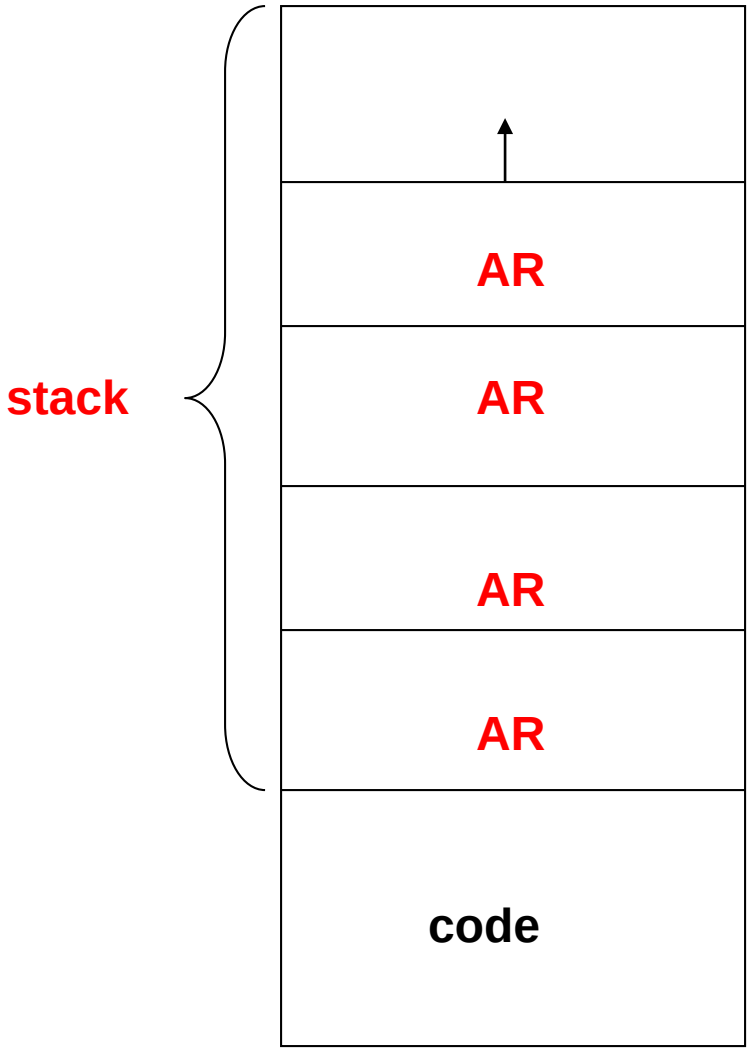
Check out Javascript example program

Virtual Machine: P- code



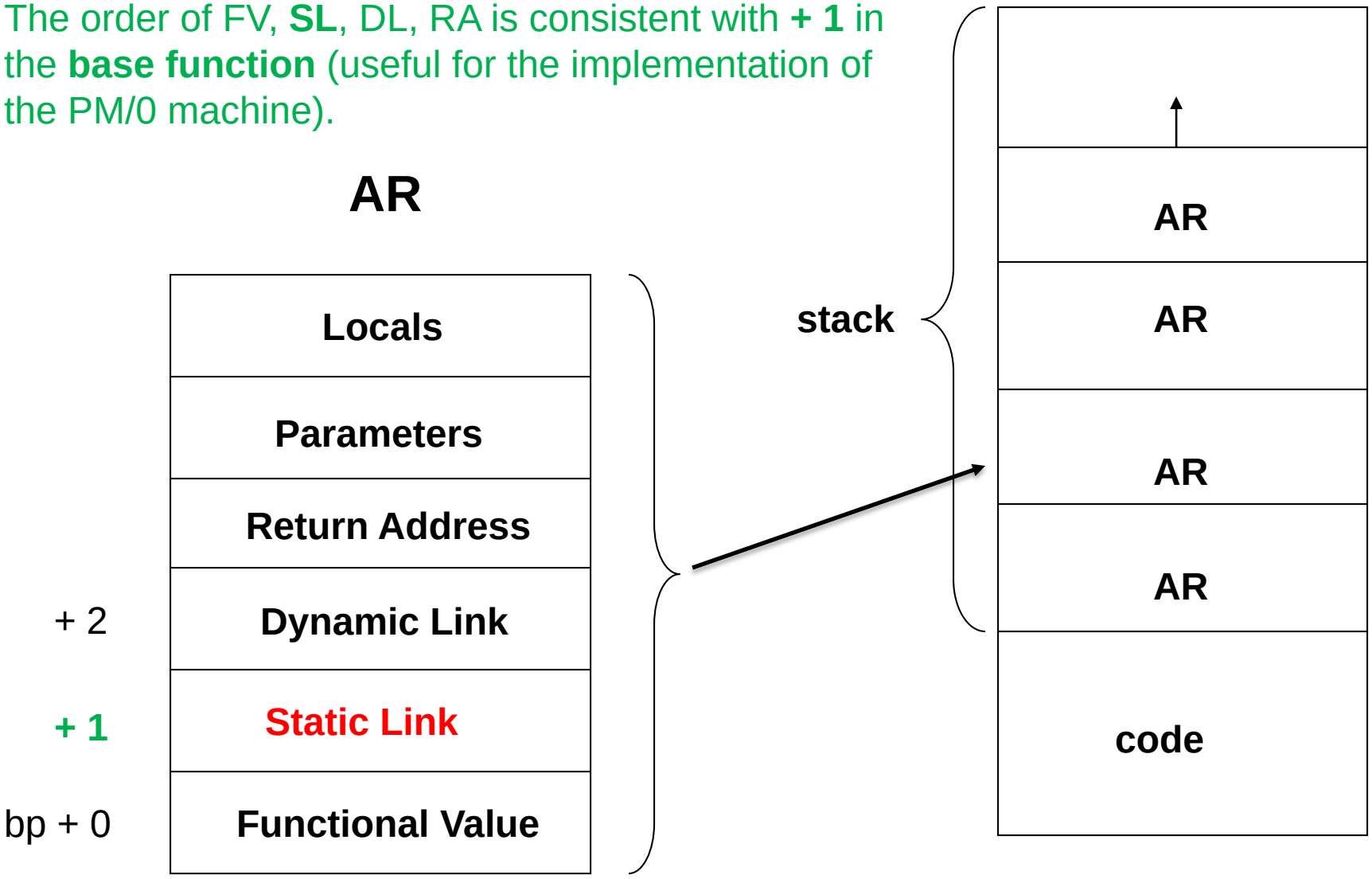
Activation Records (AR)

- **Activation Record** or **Stack Frame**: data structure that is pushed onto stack, each time a procedure/function is called
- The **dynamic links** (together with the return addresses) make it possible to return from the callee to the caller
- The **static links** allow the callee to gain access to the variables defined in its ancestor functions



Activation Records (AR)

The order of FV, **SL**, DL, RA is consistent with **+ 1** in the **base function** (useful for the implementation of the PM/0 machine).



Static Link

Static Link points to the AR of the procedure/function that statically encloses the callee.

Base function

To compute the base of the AR **L** levels below the AR whose base is b, follow the static links L times:

```
int base( int level, int b ) {
  while (level > 0) {
    b = stack[ b + 1 ];
    level--;
  }
  return b;
}
```

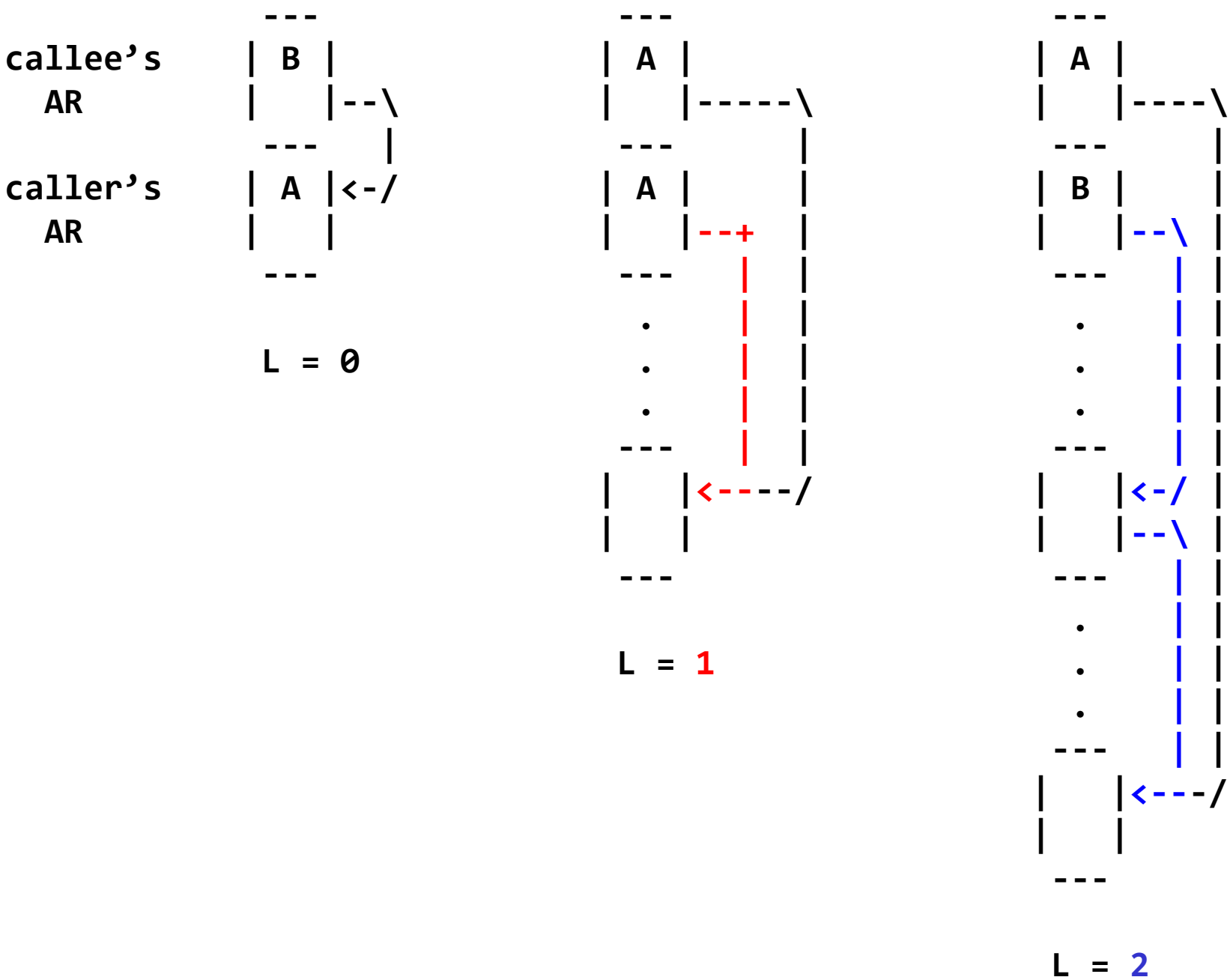
The order of FV, **SL**, DL, RA is consistent with the **+ 1** in the base function.

CAL instruction

05 **CAL L M** Call procedure at **M** (generates new stack frame)

```
stack[ sp + 2 ] ← base( L, bp ); // static link (SL)
```

- The CAL L M instruction is used to appropriately set the static link of the callee.
  - L = 0 the static link of the callee points to the base of the caller's AR
  - L = 1 the static links of the callee and the caller point to the same location
  - L > 1 depicted later
- To compile the PL/0 instruction caller-calls-callee, the compiler has to analyze the source code to determine the appropriate value for L and to emit the PM/0 instruction CAL L M


callee's AR
caller's AR
B
A
L = 0
A
A
L = 1
A
B
L = 2

Level Difference

The level difference (`ld`) between the caller and callee is defined as follows:

- a parent function calls a child function — `ld = -1`
- a function calls itself (or a sibling) — `ld = 0`
- a descendent function calls an ancestor function
  - a child calls its parent — `ld = 1`
  - a grand child calls its grandparent — `ld = 2`
  - a great-grandchild calls its great-grandparent — `ld = 3`
  - …

Formula for L in CAL instruction

To compile the PL/0 instruction caller-calls-callee, the compiler has to emit the PM/0 instruction

`CAL L M` with **`L = 1 + ld`**

- a parent function calls a child function — `L = 0`
- a function calls itself (or a sibling) — `L = 1`
- a descendent function calls an ancestor function
  - a child calls its parent — `L = 2`
  - a grand child calls its grandparent — `L = 3`
  - a great-grandchild calls its great-grandparent — `L = 4`
  - …