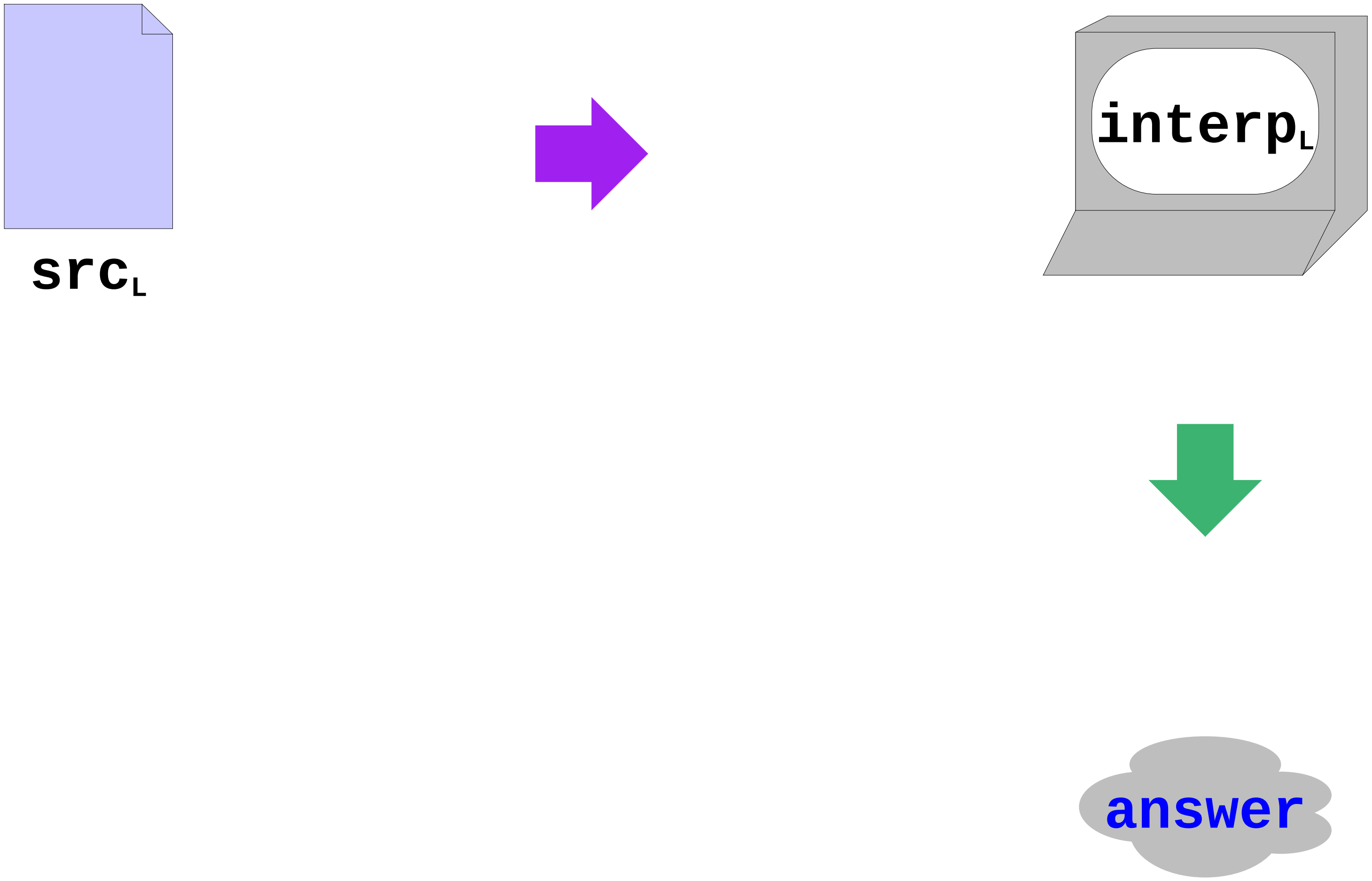
Interpreter

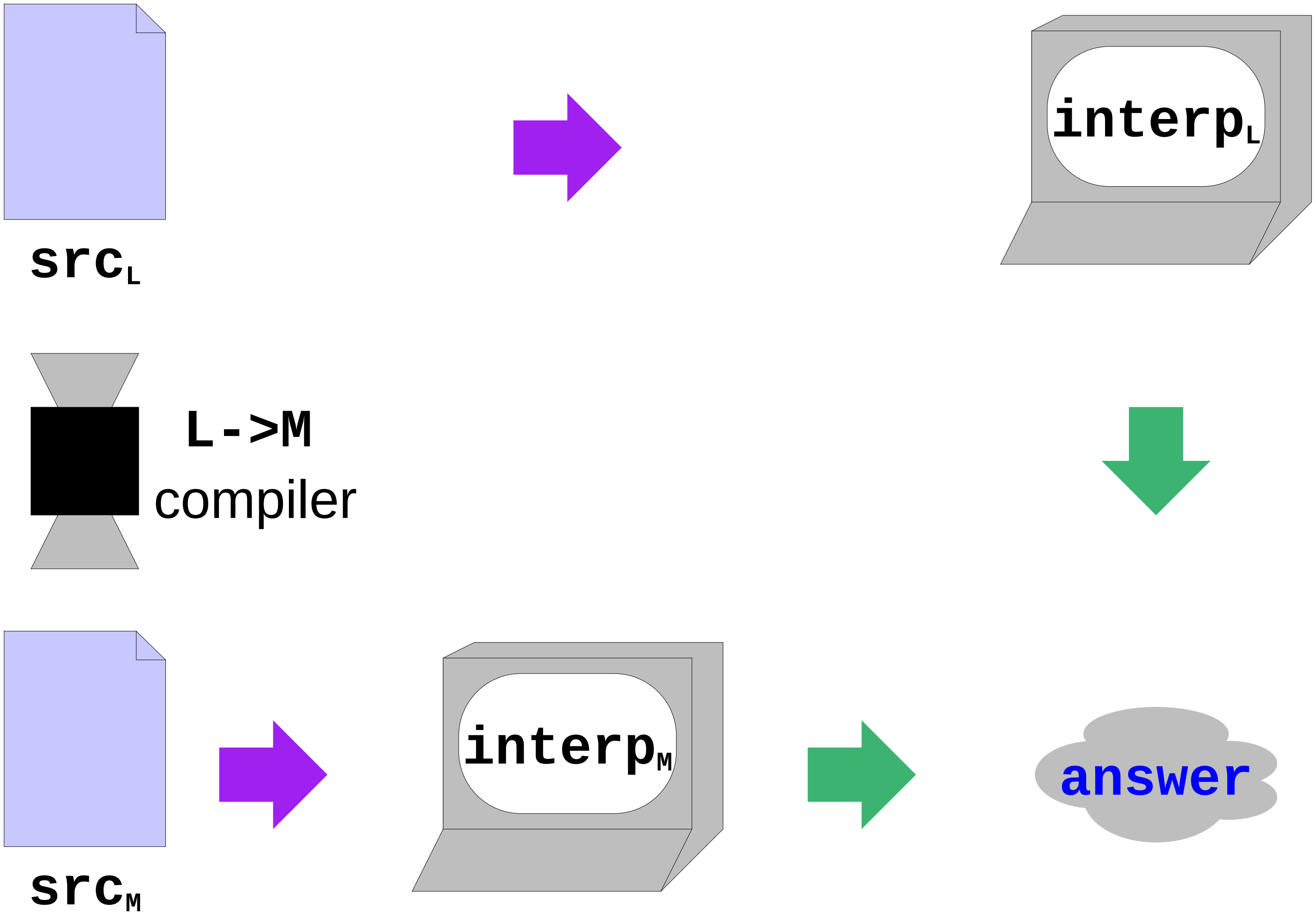


$\mathtt{src}_L$

$\mathtt{interp}_L$

answer

Compiler



src$_L$

interp$_L$

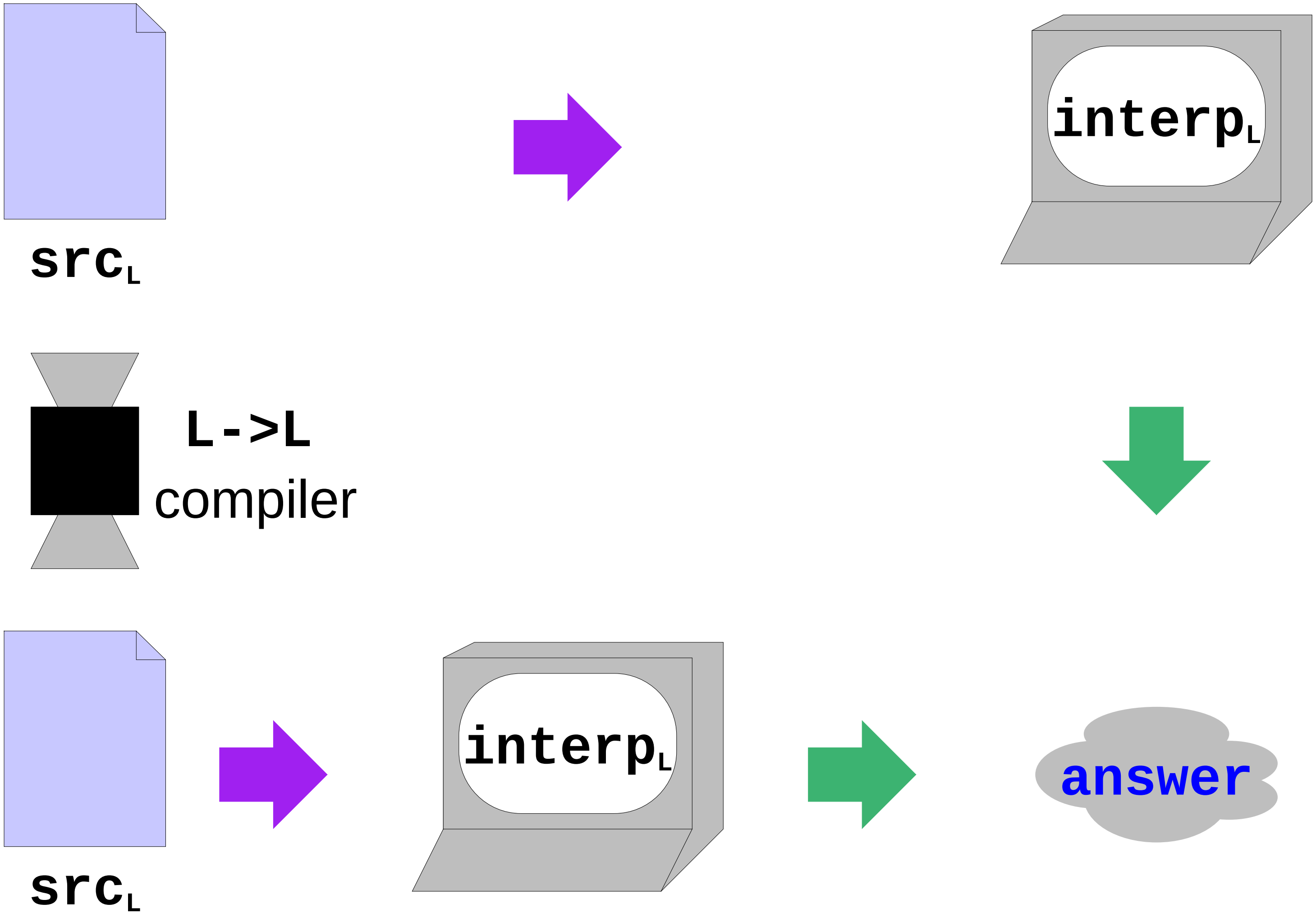
L->M
compiler

src$_M$

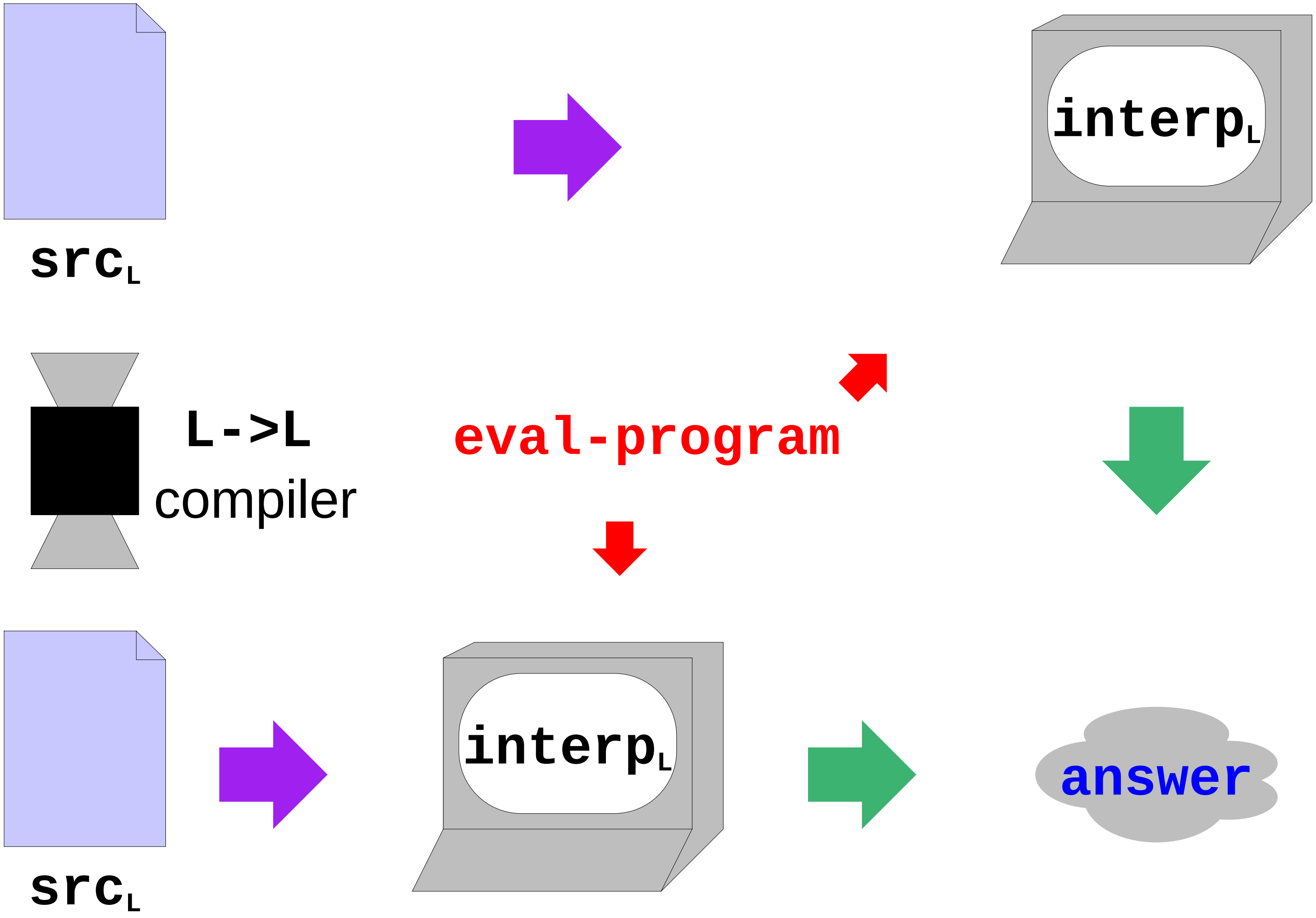
interp$_M$

answer

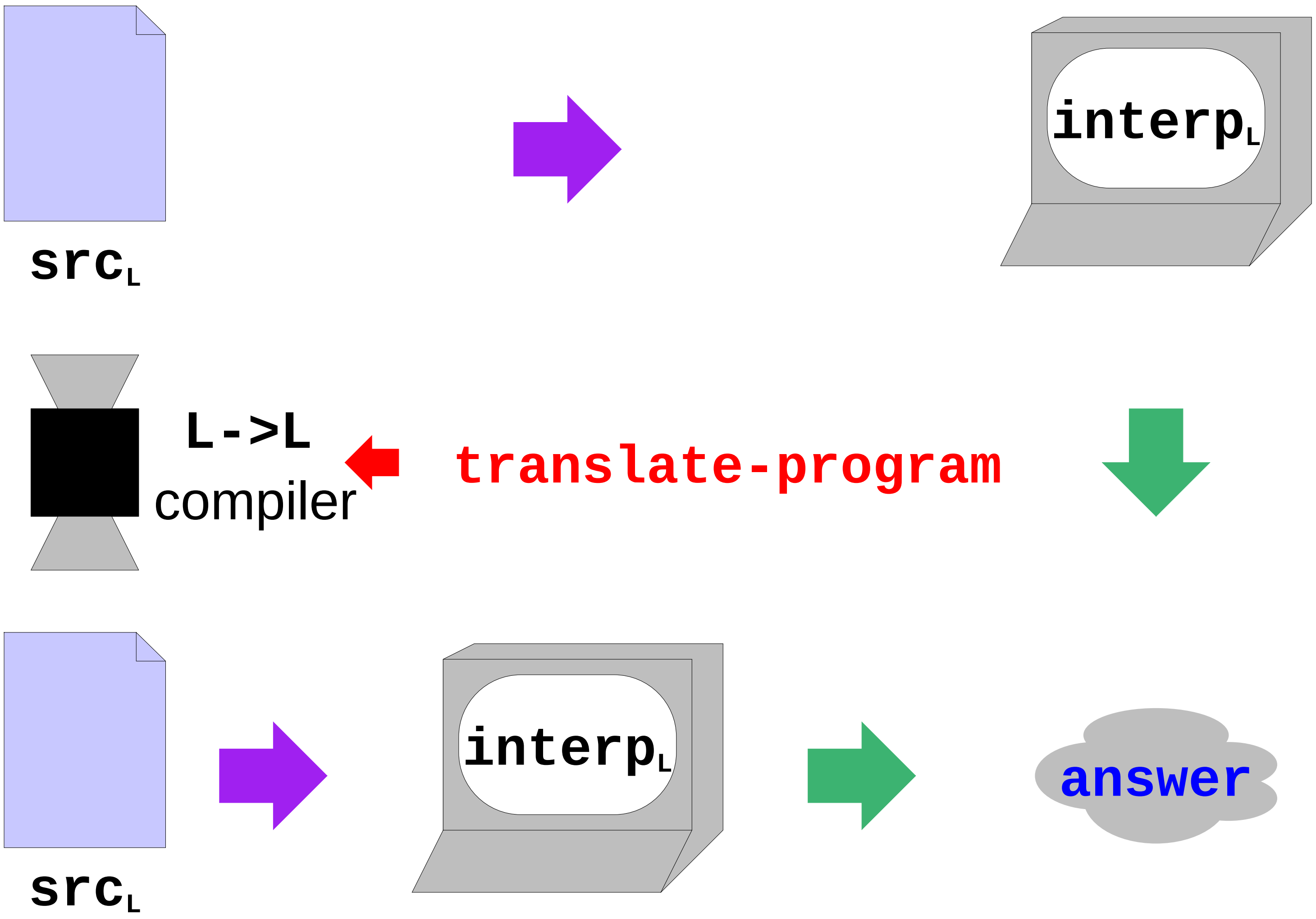
Our Compiler



Our Compiler



Our Compiler



$\mathtt{src}_L$

$\mathtt{interp}_L$

L->L

compiler

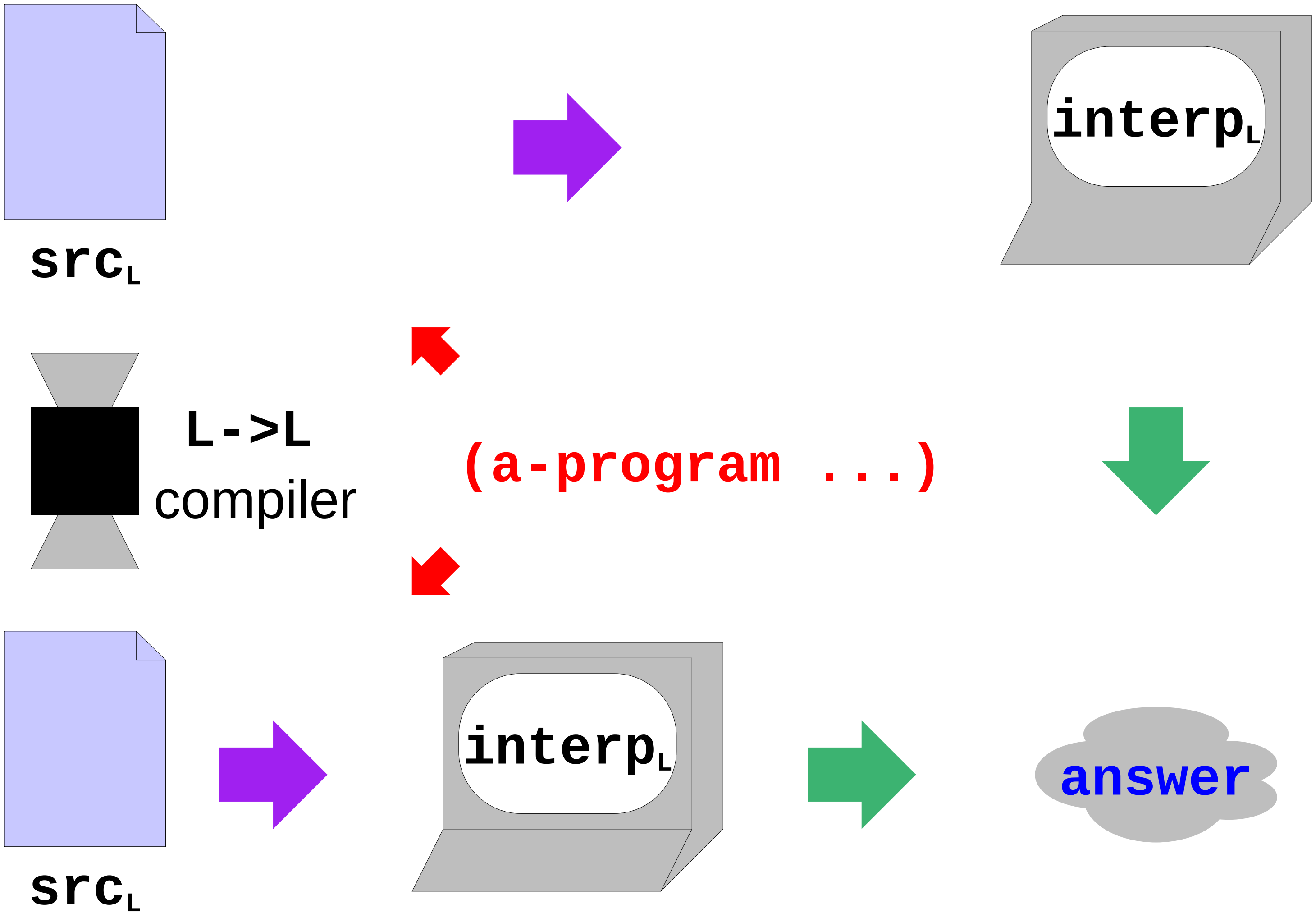
translate-program

$\mathtt{src}_L$

$\mathtt{interp}_L$

answer

Our Compiler



$\texttt{src}_L$

$\texttt{interp}_L$

L->L

compiler

(a-program ...)

$\texttt{src}_L$

$\texttt{interp}_L$

answer

Compiler: Lecture Example

- Start with simply typed **`proc`** language
- Write **`translate-expr`** that traverses the program, but makes no changes
- Add expression forms for cheaper operations
- Change **`translate-expr`** to replace expensive operations with cheaper ones
  - Lexical addresses

Compiler: HW11

- Given initial **`translate-expr`** that implements some object transformations
  - Replace method name with index in **`send`**
  - Remove obvious **`cast`** and **`instanceof`**
- Your task: add other compiler transformations
  - Lexical addresses
  - Pre-compute field array size for **`new`**
  - Pre-compute **`initialize`** index for **`new`**
  - ...