Der Code für **return** e; entspricht einer Zuweisung an eine Variable mit Relativadresse -3.

$$\text{code}\ \textbf{return}\ e;\ \rho \quad = \quad \text{code}_\text{R}\ e\ \rho$$

$$\text{storer -3}$$

$$\text{return}$$

Beispiel: Für die Funktion

```
int fac (int x) {
   if (x ≤ 0) return 1;
   else return x ∗ fac (x − 1);
}
```

erzeugen wir:

```
_fac:   enter q
        alloc 0
        loadr -3
        loadc 0
        leq
        jumpz A
        loadc 1
        storer -3
        return
        jump B
A:      loadr -3
        loadr -3
        loadc 1
        sub
        mark
        loadc _fac
        call
        slide 0
        mul
        storer -3
        return
B:      return
```

Dabei ist $\rho_{\text{fac}} : x \mapsto (L, -3)$ und $q = 1 + 1 + 3 = 5.$

10 Übersetzung ganzer Programme

Vor der Programmausführung gilt:

$$\mathrm{SP} = -1 \qquad \mathrm{FP} = \mathrm{EP} = 0 \qquad \mathrm{PC} = 0 \qquad \mathrm{NP} = \mathrm{MAX}$$

Sei $p \equiv V\_defs \quad F\_def_1 \ldots F\_def_n,$ ein Programm, wobei F_def_i eine Funktion f_i definiert, von denen eine main heißt.

Der Code für das Programm p enthält:

- Code für die Funktions-Definitionen F_def_i;
- Code zum Anlegen der globalen Variablen;
- Code für den Aufruf von main();
- die Instruktion halt.

Dann definieren wir:

$$\begin{array}{lcrl}
\text{code } p\ \emptyset & = & & \text{enter } (\text{k} + 4) \\
& & & \text{alloc } (\text{k} + 1) \\
& & & \text{mark} \\
& & & \text{loadc _main} \\
& & & \text{call} \\
& & & \text{slide k} \\
& & & \text{halt} \\
& & _\text{f}_1\text{:} & \text{code } F_def_1\ \rho \\
& & & \vdots \\
& & _\text{f}_n\text{:} & \text{code } F_def_n\ \rho
\end{array}$$

wobei $\emptyset$ $\widehat{=}$ leere Adress-Umgebung;

ρ $\widehat{=}$ globale Adress-Umgebung;

k $\widehat{=}$ Platz für globale Variablen