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

$$\begin{array}{lcl} \text{code } \textbf{return}\, e;\ \rho & = & \text{code}_R\ e\ \rho \\ & & \text{storer -3} \\ & & \text{return} \end{array}$$

Beispiel: Für die Funktion

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

erzeugen wir:

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

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

# 10 Übersetzung ganzer Programme

Vor der Programmausführung gilt:

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

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:

$$
\begin{array}{lll}
\text{code } p\ \emptyset & = & \text{enter } (\mathrm{k}+4) \\
 & & \text{alloc } (\mathrm{k}+1) \\
 & & \text{mark} \\
 & & \text{loadc \_main} \\
 & & \text{call} \\
 & & \text{slide k} \\
 & & \text{halt} \\
 & \_f_1: & \text{code } F\_def_1\ \rho \\
 & & \vdots \\
 & \_f_n: & \text{code } F\_def_n\ \rho
\end{array}
$$

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

$\rho \quad \widehat{=}$ globale Adress-Umgebung;

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