# Funktsioonid

- Olgu $f$ funktsioon, mis kutsub välja teise funktsiooni $g$.
- Funktsiooni $f$ nimetame *kutsujaks* (caller) ning funktsiooni $g$ *kutsutavaks* (callee).
- Funktsiooni väljakutse puhul emiteeritav kood tuleb ära jagada kutsuja ja kutsutava vahel.
- Täpne jaotus sõltub sellest, et kes omab millist informatsiooni.

# Funktsioonid

- Tegevused väljakutsel ja kutsutavasse sisenemisel:
  1. Registrite FP, EP salvestamine — `mark`
  2. Aktualsete argumentide arvutamine
  3. Kutsutava koodi *_g* algaadressi määramine
  4. Uue FP sättimine — `call`
  5. PC salvestamine ja *_g* algusse hüppamine
  6. Uue EP sättimine — `enter`
  7. Lokaalsetele muutujatele mälu eraldamine — `alloc`
- Tegevused kutsutavast lahkumisel:
  1. Registrite FP, EP, SP taastamine — `return`
  2. Tagasipöördumine *f*-i koodi; so. registri PC taastamine

# Funktsioonid

$$
\begin{aligned}
\text{code}_R\ (g\,(e_1,\ldots,e_n))\ \rho \quad = \quad & \texttt{mark} \\
& \text{code}_R\ e_1\ \rho \\
& \ldots \\
& \text{code}_R\ e_n\ \rho \\
& \text{code}_R\ g\ \rho \\
& \texttt{call n}
\end{aligned}
$$

- Aktuaalsete parameetritena esinevatel avaldistel leitakse nende R-väärtused
  - Parameetrite edastamine *väärtuse* kaudu (call-by-value)
- Funktsioon $g$ võib olla avaldis, mille R-väärtus on kutsutava funktsiooni algaadress

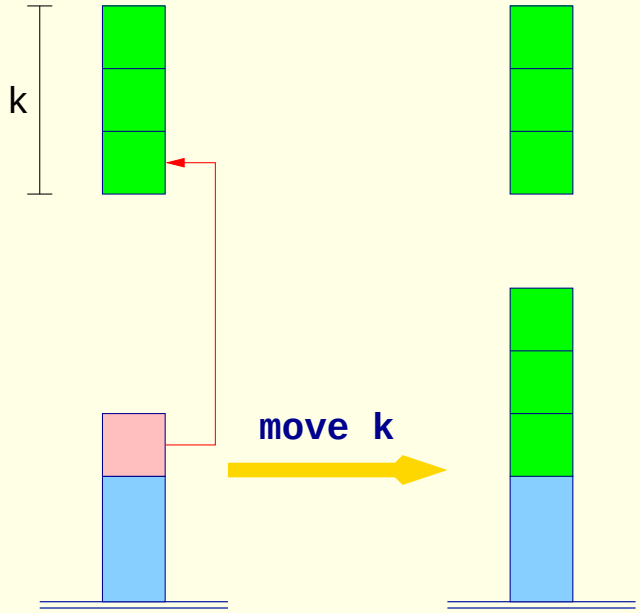
# Funktsioonid

- Funktsiooni nimi on *viitkonstant*, mille R-väärtus on funktsiooni algaadress
- Funktsiooni viida kaudu väärtuse võtmine väljastab selle sama viida
  - Näide: deklaratsiooni **int** $(*)()g$; korral on väljakutsed $g()$ ja $(*g)()$ ekvivalentsed
- Argumentidena antavad kirjed kopeeritakse

| | | | |
|---|---|---|---|
| $\text{code}_R\ f\ \rho$ | = | $\rho\ f$ | $f$ on funktsiooni nimi |
| $\text{code}_R\ (*e)\ \rho$ | = | $\text{code}_R\ e\ \rho$ | $e$ on funktsiooni viit |
| $\text{code}_R\ e\ \rho$ | = | $\text{code}_L\ e\ \rho$ | |
| | | `move k` | $e$ on kirje suurusega $k$ |

# Funktsioonid

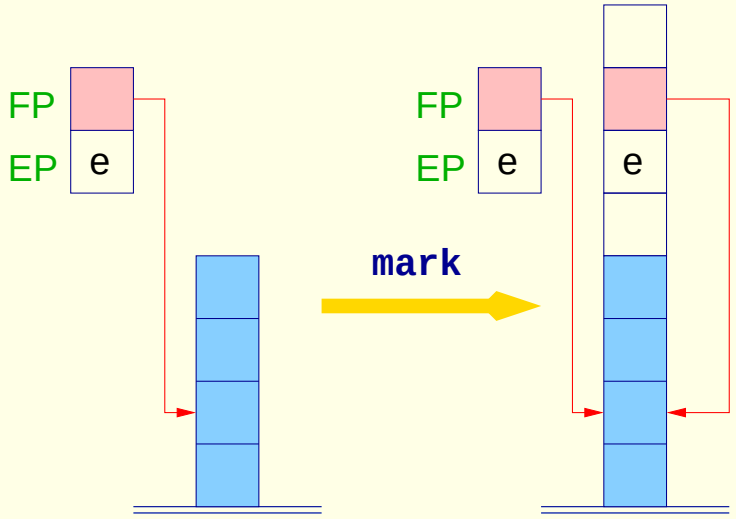
Käsk `move k` kopeerib *k* pesa magasini tippu




```
for (i=k-1; i ≥ 0; i–)
    S[SP+i] = S[S[SP]+i];
SP = SP+k−1;
```

# Funktsioonid

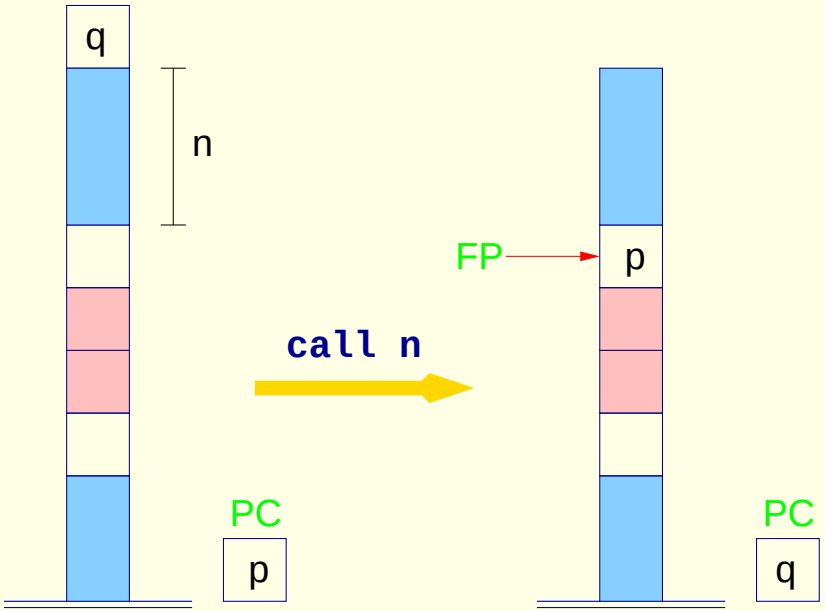
Käsk `mark` reserveerib mälu organisatoorsete pesade ja resultaatväärtuse jaoks ning salvestab registrid FP ja EP




```
S[SP+2] = EP;
S[SP+3] = FP;
SP = SP + 4;
```

# Funktsioonid

Käsk `call n` salvestab jätkuaadressi ning omistab registritele FP, SP ja PC uued väärtused




```
FP = SP - n - 1;
S[FP] = PC;
PC = S[SP];
SP−−;
```

# Funktsioonid

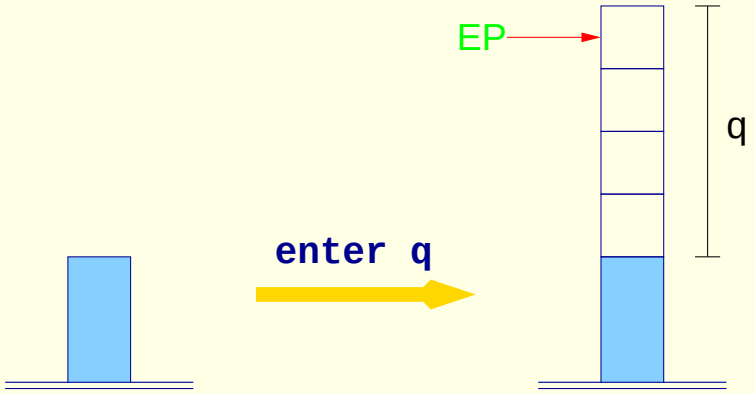
$$\text{code}\ (t\ f\ (args)\{vars\ ss\})\ \rho \quad = \quad \_f\texttt{:}\ \texttt{enter q}$$
$$\texttt{alloc k}$$
$$\text{code}\ ss\ \rho_f$$
$$\texttt{return}$$

kus

- $\texttt{q} = maxS + \texttt{k}$
- $maxS$ = lokaalse magasini maksimaalne sügavus
- $\texttt{k}$ = mälu lokaalsetele muutujatele
- $\rho_f$ = $f$-i aadresskeskkond

# Funktsioonid

Käsk `enter q` seab registrile EP uue väärtuse

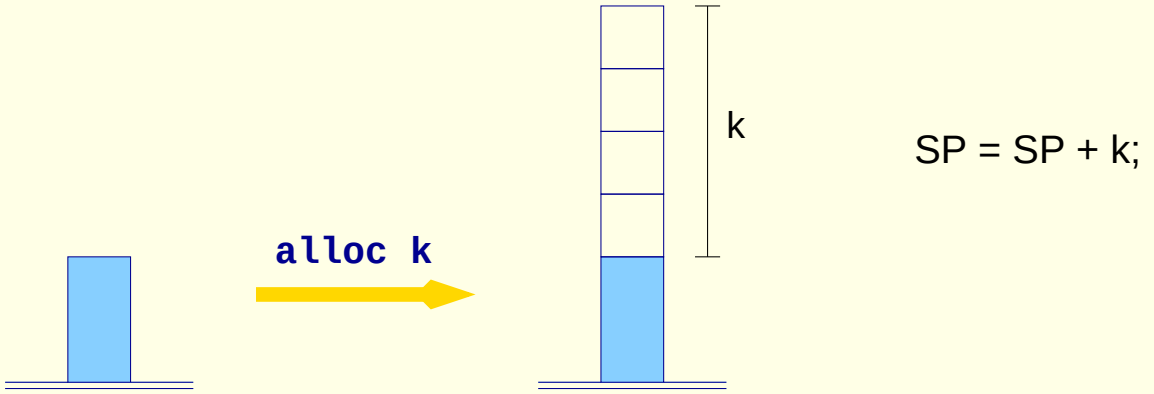



```
EP = SP + q;
if (EP ≥ NP)
    Error ("Stack Overflow");
```

NB! Kui mälu pole piisavalt, siis programmi töö katkestatakse

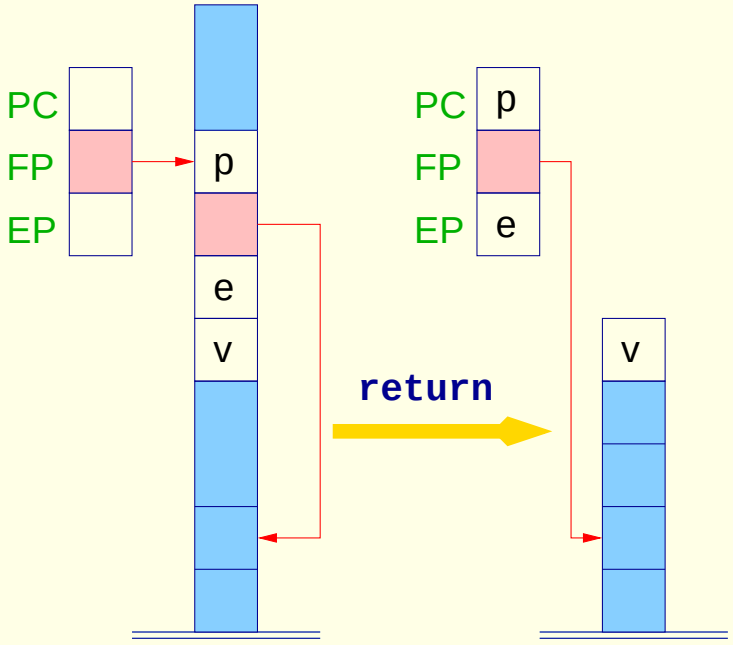
# Funktsioonid

Käsk `alloc k` reserveerib magasinis mälu lokaalsete muutujate jaoks



# Funktsioonid

Käsk `return` taastab registrite PC, FP ja EP sisu ning jätab resultaatväärtuse magasini tippu



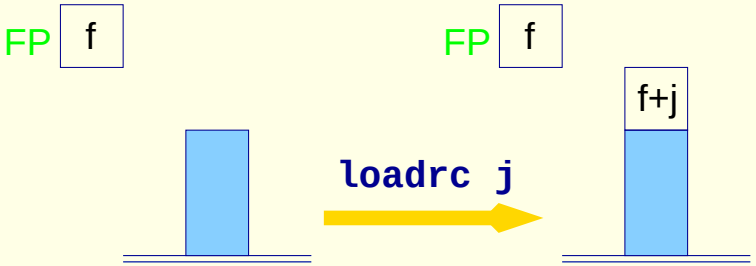

```
PC = S[FP];
EP = S[FP−2];
if (EP ≥ NP)
    Error ("Stack Overflow");
SP = FP − 3;
FP = S[SP+2];
```

# Funktsioonid

Lokaalsetele muutujatele ja formaalsetele parameetritele ligipääs toimub relatiivselt registri FP suhtes

$$\text{code}_L\ x\ \rho \quad = \quad \begin{cases} \texttt{loadc j} & \text{kui } \rho\ x = (G, j) \\ \texttt{loadrc j} & \text{kui } \rho\ x = (L, j) \end{cases}$$

Käsk `loadrc j` leiab FP ja `j` summa




```
SP++;
S[SP] = FP + j;
```

# Funktsioonid

Analoogselt käskudega `loada j` ja `storea j` toome sisse käsud `loadr j` ja `storer j`

$$
\begin{array}{lcl}
\texttt{loadr j} & = & \texttt{loadrc j} \\
 & & \texttt{load} \\
\texttt{storer j} & = & \texttt{loadrc j} \\
 & & \texttt{store}
\end{array}
$$

return-lausele vastab väärtuse omistamine muutujale suhtaadressiga $-3$

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

# Funktsioonid

Näide:

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

Siis $\rho_{\text{fac}} : x \mapsto (L, 1)$ ja emiteeritav kood on:

```
_fac:  enter 7      loadc 1      A:  loadr 1          mul
       alloc 0      storer -3        mark             storer -3
       loadr 1      return           loadr 1          return
       loadc 0      jump B           loadc 1     B:   return
       leq                           sub
       jumpz A                       loadc _fac
                                     call 1
```

# Kogu programmi transleerimine

Abstraktse masina olek enne programmi käivitamist:

SP = -1 FP = EP = 0 PC = 0 NP = MAX

Olgu $p \equiv vars\ fdef_1 \ldots fdef_n$, kus $fdef_i$ on funktsiooni $f_i$ ja milledest üks on nimega main.
Emiteeritav kood koosneb:

- funktsioonide definitsioonidele $fdef_i$ vastav kood;
- globaalsetele muutujatele mälu reserveerimine;
- funktsiooni main(); väljakutse kood;
- käsk `halt`.

# Kogu programmi transleerimine

$$
\begin{array}{lll}
\text{code } p\,\emptyset = & \texttt{enter}\ (\texttt{k}+6) & \texttt{pop} \\
 & \texttt{alloc}\ (\texttt{k}+1) & \texttt{halt} \\
 & \texttt{mark} & \_f_1 : \text{code } fdef_1\ \rho \\
 & \texttt{loadc \_main} & \vdots \\
 & \texttt{call 0} & \_f_n : \text{code } fdef_n\ \rho
\end{array}
$$

kus $\emptyset$ = tühi aadresskeskkond

$\rho$ = globaalne aadresskeskkond

$\texttt{k}$ = mälu globaalsetele muutujatele