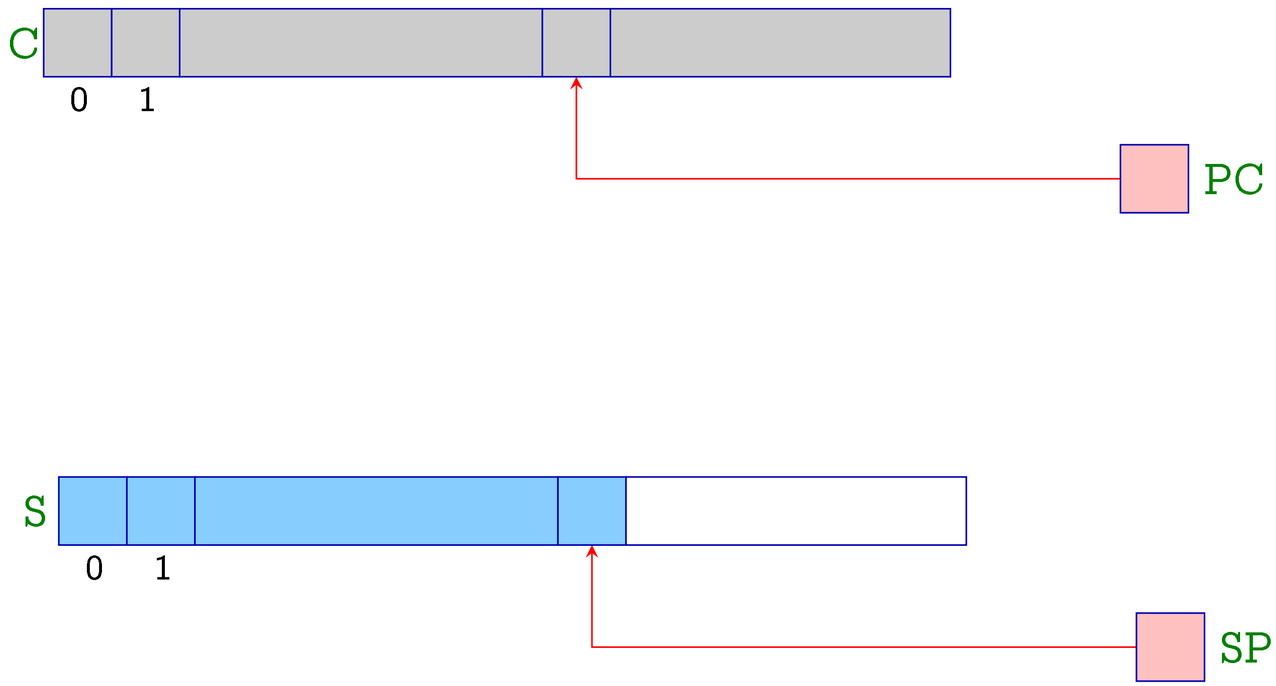
CMa - lihtsustatud C abstraktne masin

CMa arhitektuur
Avaldiste transleerimine
Lausete transleerimine

Abstraktse masina kirjeldus

- Abstraktne masin määrab hulga käske,
  mis on täitmiseks abstraktsel riistvaral.
- Abstraktse riistavara kirjeldus:
  andmestruktuurid ja kuidas käsud neid kasutavad.
- Protsessi juhib käituskeskkond (*run-time system*).

Andmestruktuurid



- C (*code store*). Massiivi igas pesas on käsk.
- PC (*program counter*). Programmiloendur viitab järgmisele käsule.
- S (*Stack*). Magasin andmete hoidmiseks.
- SP (*Stack pointer*). Magasiniviit ülemise elemendi aadressile.

Programmi täitmine

- Masin laadib käsu C[PC] käsu-registrisse IR (*instruction register*).
- Seejärel suurendatakse PC ühe võrra
- ja lõpuks täidetakse käsk.

```
while (true) {
      IR = C[PC];
      PC++;
      execute(IR);
}
```

- Käsu täitmisel võib muuta PC (*jump*) ja programmi töö lõpetada (*halt*).

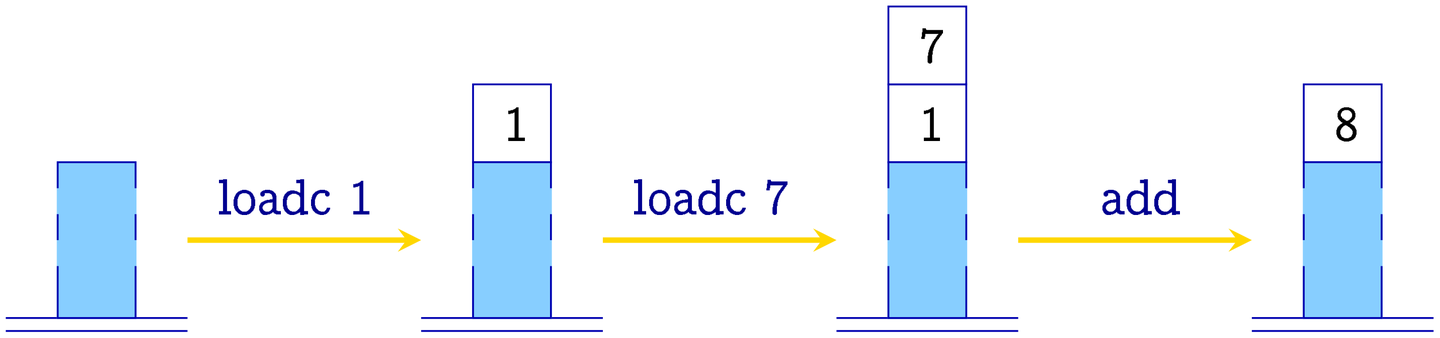
Aritmeetikatehted	`add, sub, mul, div, mod`
Loogikatehted	`and, or, xor, not`
Võrdlustehted	`eq, neq, le, leq, ge, geq`
Mäluoperatsioonid	`loadc, load, store, alloc`
Juhtimiskäsud	`jump, jumpz, halt`

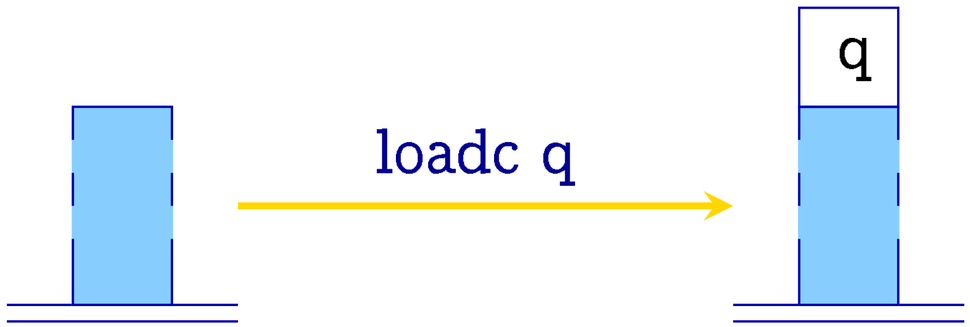
Põhikäsud

Courses lehel on terviklik CMa käsustik!

Magasini peal arvutamine

- Kui tahame arvutada näiteks avaldist $1 + 7$, siis …
- Arvutame alamavaldised 1 ja 7 ning paneme need magasini tippu.
- Rakendame käsku `add`, mis vastab operaatorile $+$

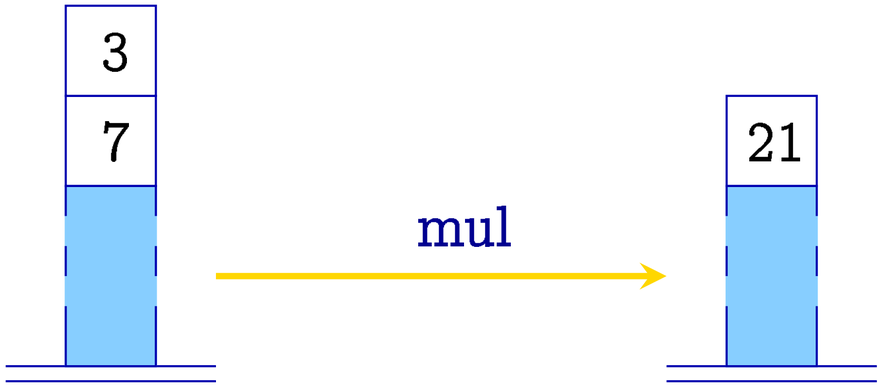





```
SP++;
S[SP] = q;
```

`loadc` q

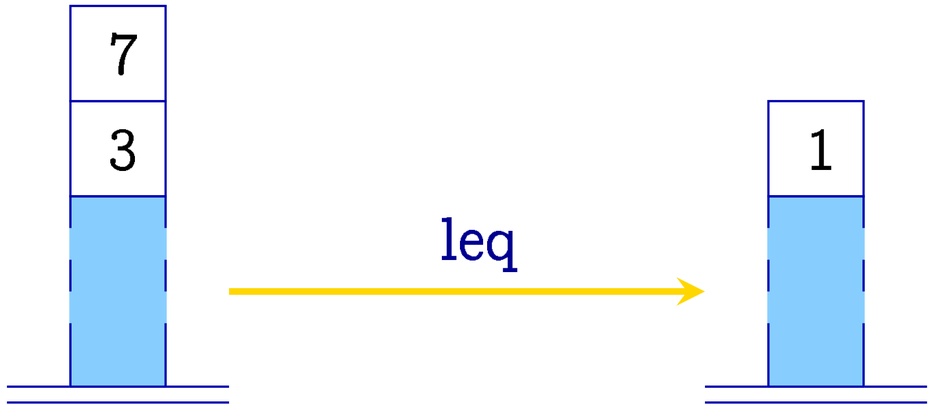
Salvestab konstandi q magasini tippu.




```
SP--;
S[SP] = S[SP] * S[SP+1];
```

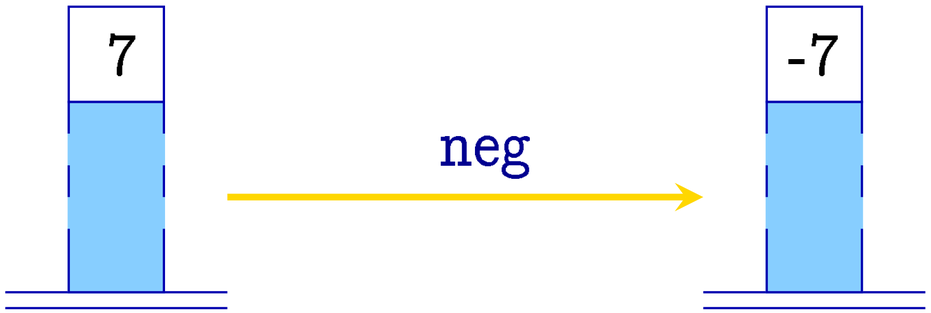
Binaarsed operatsioonid

Tarvitavad kaks argumenti magasini tipust ja salvestavad tulemuse tagasi.
`loadc` 7; `loadc` 3; `sub` korral arvutab $7 - 3$.

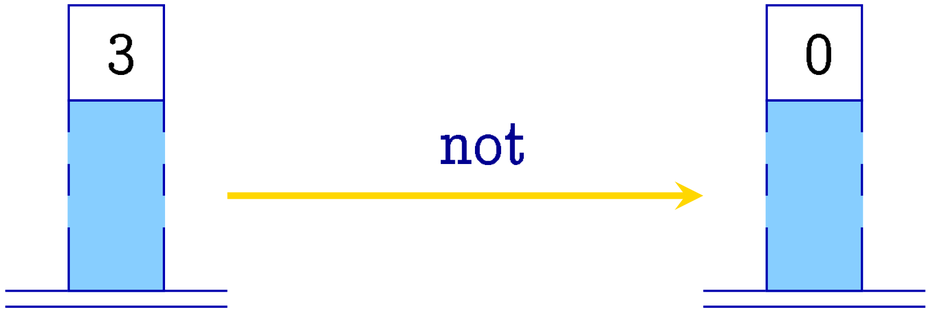


Võrdlusoperatsioonid

Salvestavad 1, kui võrdlus lehtib, ja 0, kui ei kehti.




```
S[SP] = -S[SP];
```



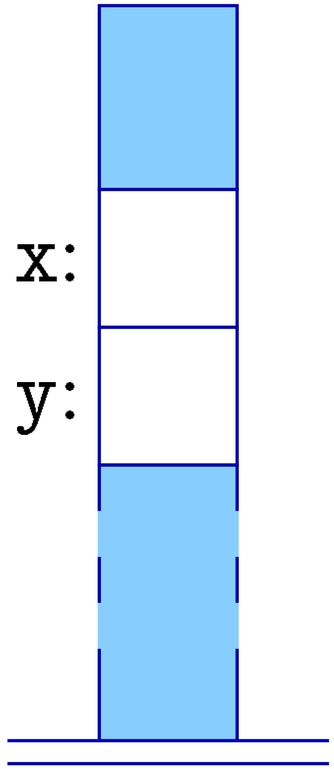

```
if (S[SP] ≠ 0)
    S[SP] = 0;
else
    S[SP] = 1;
```

Unaarsed operatsioonid

(`not` on loogiline eitus, mitte bithaaval.)

Muutujad!

- Muutujate väärtuseid hoiame ka magasinis $\mathtt{S}$.
- Programmi deklaratsioonide põhjal loome aadresskeskkonna.
- Näiteks, kui aadresskeskkond on $\rho = \{x \mapsto 5, y \mapsto 4\}$, siis teame, et muutuja x väärtus asub pesas $\mathtt{S}[5]$.
- Eeldame siin, et aadresskeskkond on fikseeritud.



L ja R väärtused

- Tähtis Tõde: Muutujaid kasutatakse kahel erineval moel!
- Näiteks omistuslauses $x = y + 1$ oleme huvitatud
  - muutuja y väärtusest, kuid
  - muutuja x aadressist.
- Muutuja kasutuse süntaktilisest asukohast sõltuvalt vajame, kas tema L-väärtust (aadress) või R-väärtust (sisu).

Koodi genereerimine

- Funktsioon $\texttt{code}_{\texttt{L}}$ loob L-väärtust arvutava koodi.
- Funktsioon $\texttt{code}_{\texttt{R}}$ teeb sama R-väärtuse jaoks.
- NB! Mitte igal avaldisel pole L-väärtust (näiteks $x + 1$).

R-väärtuste arvutamine

- Binaarse operatsiooni korral arvutame argumentide väärtused ja rakendame nendel operatsioonile vastavat käsku:

$$\begin{aligned} \texttt{code}_R\ (e_1 + e_2)\ \rho &= \texttt{code}_R\ e_1\ \rho \\ &\quad\ \texttt{code}_R\ e_2\ \rho \\ &\quad\ \texttt{add} \end{aligned}$$

- Unaarsete operatsioonidega samamoodi:

$$\begin{aligned} \texttt{code}_R\ (-e)\ \rho &= \texttt{code}_R\ e\ \rho \\ &\quad\ \texttt{neg} \end{aligned}$$

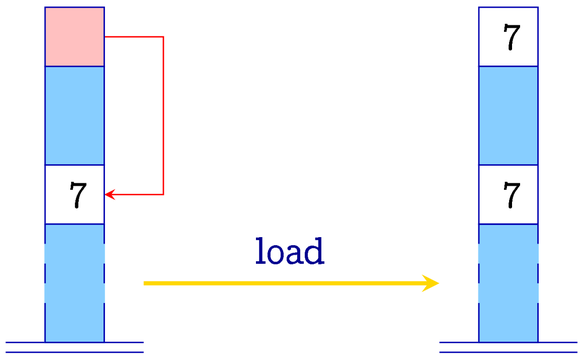
- Ja konstantide jaoks oli käsk `loadc`:

$$\texttt{code}_R\ q\ \rho = \texttt{loadc q}$$

Muutujate transleerimine

- Muutuja L-väärtus on kirjas aadresskeskkonnas ρ.
- Muutuja R-väärus on L-väärtuse pesas olev sisu!
- Selle saab kätte käsuga `load`.

$$\begin{aligned} \texttt{code}_L\ x\ \rho &= \texttt{loadc}\ (\rho\ x) \\ \texttt{code}_R\ x\ \rho &= \texttt{code}_L\ x\ \rho \\ &\quad\ \texttt{load} \end{aligned}$$

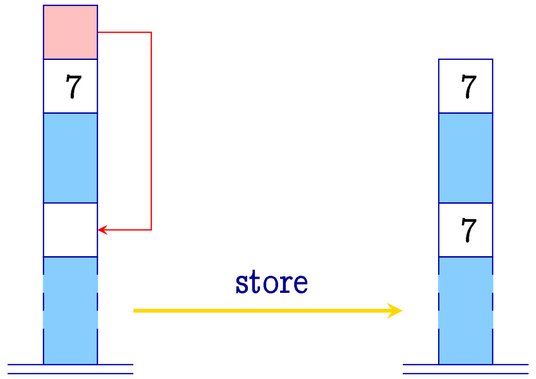



```
S[SP] = S[S[SP]];
```

Omistuse transleerimine

- Parema poole R-väärtus ja vasaku poole L-väärtus!
- Magasini tipus on mälupesa, kuhu tuleks salvestada.
- Väärtuse salvestab pessa `store`.

$$\texttt{code}_R\ (x = e)\ \rho \quad = \quad \begin{array}{l} \texttt{code}_R\ e\ \rho \\ \texttt{code}_L\ x\ \rho \\ \texttt{store} \end{array}$$




```
S[S[SP]] = S[SP-1];
SP--;
```

Näide: olgu $e \equiv (x = y - 1)$ and $\rho = \{x \mapsto 4, y \mapsto 7\}$,
siis $\texttt{code}_R\ e\ \rho$ emiteerib koodi:

```
loadc 7        sub
load           loadc 4
loadc 1        store
```

$\texttt{code}_R\ (y-1)\ \rho$
$\texttt{code}_L\ x\ \rho$
`store`

$\Longrightarrow$

$\texttt{code}_R\ y\ \rho$
$\texttt{code}_R\ 1\ \rho$
`sub`
$\texttt{code}_L\ x\ \rho$
`store`

$\Longrightarrow$

`loadc` 7
`load`
$\texttt{code}_R\ 1\ \rho$
`sub`
$\texttt{code}_L\ x\ \rho$
`store`

$\Longrightarrow$

`loadc` 7
`load`
`loadc` 1
`sub`
$\texttt{code}_L\ x\ \rho$
`store`

$\Longrightarrow$

`loadc` 7
`load`
`loadc` 1
`sub`
`loadc` 4
`store`

Miks nii keeruline?

- Tegelikult saab lihtsustada.
- Aga L-väärtus ei pea olema ainult muutuja, näiteks $x[i+1]$.
- L-väärtuse arvutamiseks on vaja avaldise $i+1$ väärtus arvutada:

$$\texttt{code}_L\ (x[i+1])\ \rho = \texttt{code}_L\ x\ \rho$$
$$\texttt{code}_R\ (i+1)\ \rho$$
$$\texttt{add}$$

```
loada q   =   loadc q
              load

storea q  =   loadc q
              store
```

```
loadc 7                 loada 7
load                    loadc 1
loadc 1        ⟹       sub
sub                     storea 4
loadc 4
store
```

Nüüd laused

- Tähtis Tõde #2: Avaldisel on väärtus; lausel on vaid kõrvalmõju.
- Omistamine on Javas avaldis, seega saab kirjutada $x = y = 0;$
- Kui $x = y$ on avaldis, millel on väärtus, siis $x = y;$ on lause, kus tulemus tuleb ära unustada.


```
SP--;
```

Lausete transleerimine

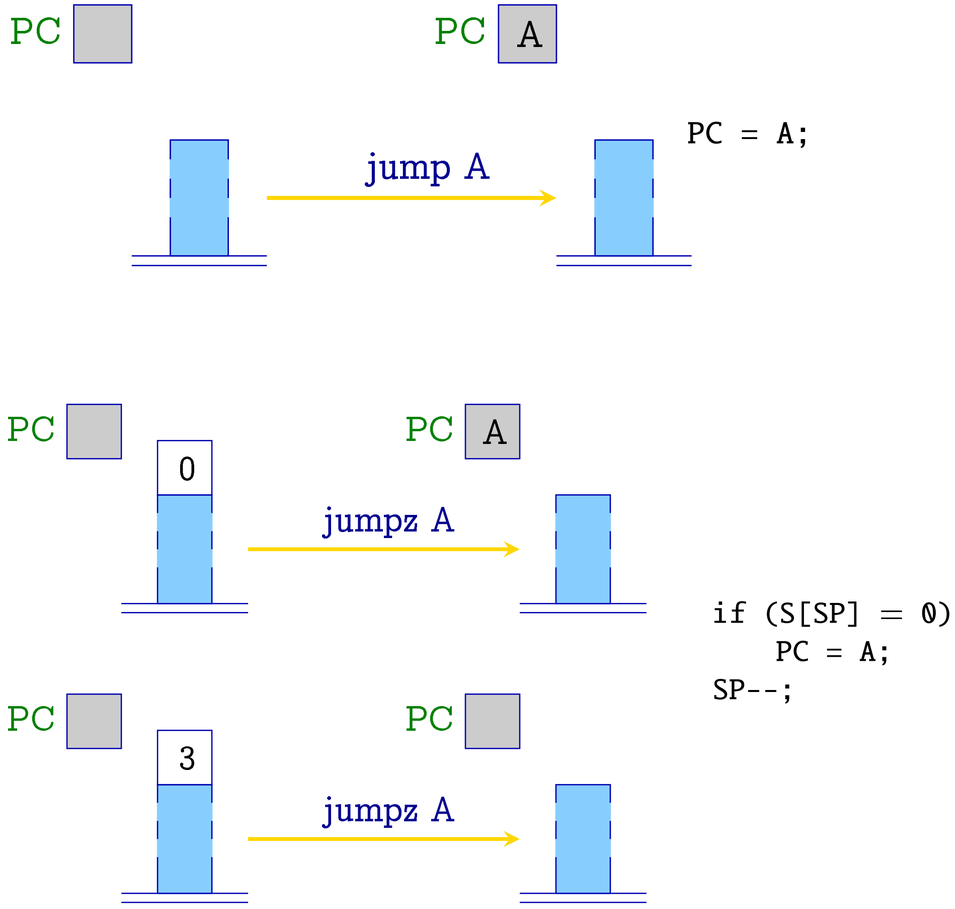
- Omistuslausele vastav kood on R-väärtuse kaudu defineeritud.
- Tulemus eemaldatakse käsuga `pop`.
- Lausete jada korral on vaja ainult käske järjest tõlkida.

$$
\begin{aligned}
\texttt{code}\ (e;)\ \rho &= \texttt{code}_R\ e\ \rho \\
&\quad\ \texttt{pop} \\
\texttt{code}\ (s\ ss)\ \rho &= \texttt{code}\ s\ \rho \\
&\quad\ \texttt{code}\ ss\ \rho \\
\texttt{code}\ \varepsilon\ \rho &=
\end{aligned}
$$

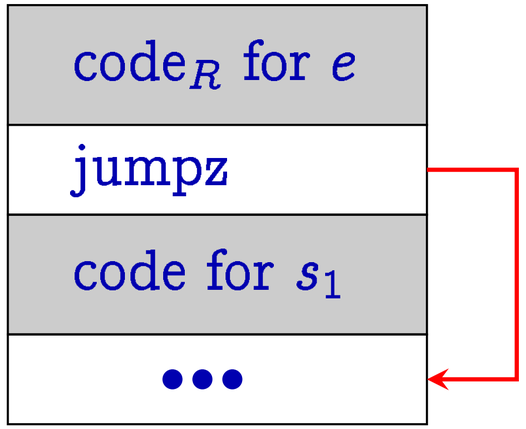
```
while (true) {
    IR = C[PC];
    PC++;
    execute(IR);
}
```

Hüppamine!

- Käituskeskkond suurendab käsuloendurit, seega on valmis järgmist käsku täitma.
- Tingimuslausete korral on vaja hüpata. Selleks kasutame sümbolmärgendeid.
- Käsk `jump` A hüpab märgendi A juurde.
- Käsk `jumpz` A hüpab ainult siis, kui magasini tipus on arv 0.



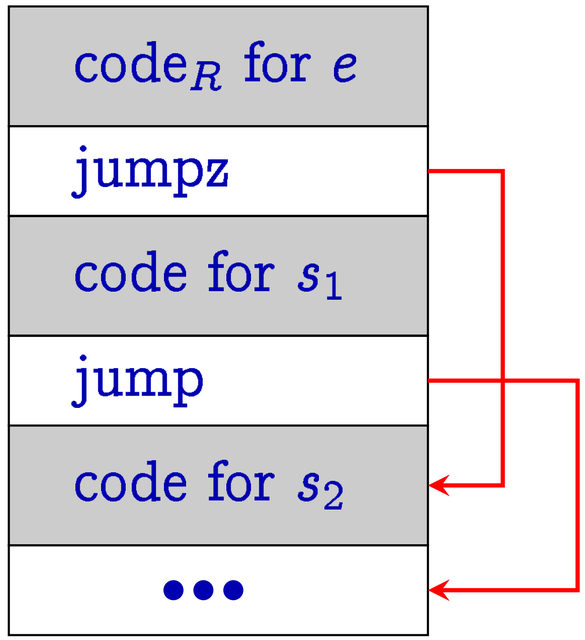
$$
\begin{aligned}
\texttt{code}\ (\textbf{if}\ (e)\ s_1)\ \rho\ \ = & \\
& \texttt{code}_R\ e\ \rho \\
& \texttt{jumpz}\ \ \texttt{A} \\
& \texttt{code}\ s_1\ \rho \\
\texttt{A:}\ & \ldots
\end{aligned}
$$



Tingimuslause transleerimine

Kui tingimus ei kehti, tuleb s_1 koodist üle hüpata.

$$
\begin{aligned}
&\text{code } (\textbf{if } (e)\ s_1\ \textbf{else}\ s_2)\ \rho \quad = \\
&\qquad\qquad\qquad\qquad \text{code}_R\ e\ \rho \\
&\qquad\qquad\qquad\qquad \texttt{jumpz A} \\
&\qquad\qquad\qquad\qquad \text{code}\ s_1\ \rho \\
&\qquad\qquad\qquad\qquad \texttt{jump B} \\
&\qquad\qquad\qquad \texttt{A:}\ \text{code}\ s_2\ \rho \\
&\qquad\qquad\qquad \texttt{B:}\ \ldots
\end{aligned}
$$



Tingimuslause else-haruga

Kui tingimus ei kehti, tuleb hüpata s_2 koodi juurde.
Kui tingimus kehtib, peab aga s_1 lõpus s_2 koodist üle hüppama!

Näide: olgu $\rho = \{x \mapsto 4, y \mapsto 7\}$ ja

$$
\begin{array}{rcll}
s & \equiv & \textbf{if } (x > y) & (i) \\
 & & \quad x = x - y; & (ii) \\
 & & \textbf{else } y = y - x; & (iii)
\end{array}
$$

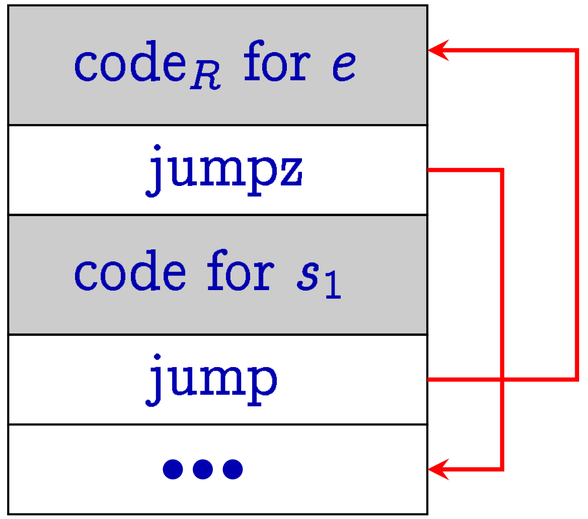
siis code s ρ emiteerib koodi:

```
loada 4
loada 7
ge
jumpz A
```
(i)

```
loada 4
loada 7
sub
storea 4
pop
jump B
```
(ii)

```
A: loada 7
   loada 4
   sub
   storea 7
   pop
B: ...
```
(iii)

$$
\begin{aligned}
&\text{code } (\textbf{while } (e)\ s_1)\ \rho \quad = \\
&\qquad\qquad \text{A: code}_R\ e\ \rho \\
&\qquad\qquad\quad \texttt{jumpz B} \\
&\qquad\qquad\quad \text{code } s_1\ \rho \\
&\qquad\qquad\quad \texttt{jump A} \\
&\qquad\qquad \text{B: } \ldots
\end{aligned}
$$



While-tsükli transleerimine

Välja hüpata, kui tingimus ei kehti.
Tsükli keha lõpus hüpata tagasi tsükli pea juurde.

Näide: olgu $\rho = \{a \mapsto 7, b \mapsto 8, c \mapsto 9\}$ ja

$$
\begin{array}{rcll}
s & \equiv & \textbf{while}\ (a > 0)\ \{ & (i) \\
 & & \quad c = c + 1; & (ii) \\
 & & \quad a = a - b; & (iii) \\
 & & \} &
\end{array}
$$

siis code s ρ emiteerib koodi:

```
A: loada 7      loada 9      loada 7        jump A
   loadc 0      loadc 1      loada 8     B: ...
   ge           add          sub
   jumpz B      storea 9     storea 7
                pop          pop
```

(i) (ii) (iii)

Iseseisvalt: kuidas teha for-tsükkel?

Kokkuvõte

- CMa abstraktse masina tööpõhimõte: magasini peal arvutamine.
- Avaldiste ja lausete süntaksjuhitav transleerimine.
- Avaldiste L-väärtus ja R-väärtus.