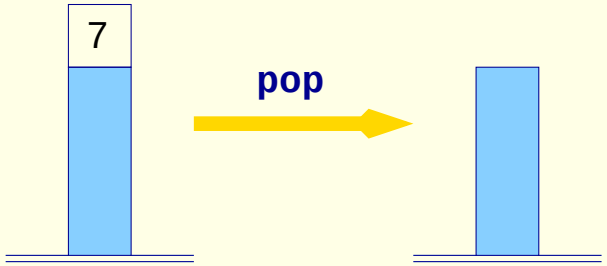
Laused ja lausete jadad

- Kui *e* on avaldis, siis *e*; on lause.
- Lausel ei ole väärtust; seega registri SP sisu peab enne ja pärast lausele vastava koodi täitmist olema sama.

$$
\begin{aligned}
\text{code}\ (e;)\ \rho &= \text{code}_R\ e\ \rho \\
&\quad\ \texttt{pop} \\
\text{code}\ (s\ ss)\ \rho &= \text{code}\ s\ \rho \\
&\quad\ \text{code}\ ss\ \rho \\
\text{code}\ \varepsilon\ \rho &= \qquad\qquad \text{// tühi käskude jada}
\end{aligned}
$$

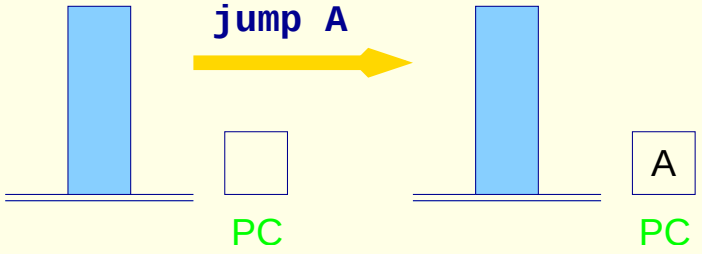
- Käsk pop eemaldab magasinist kõige ülemise pesa:



SP−−

Tingimuslikud laused ja tsüklid

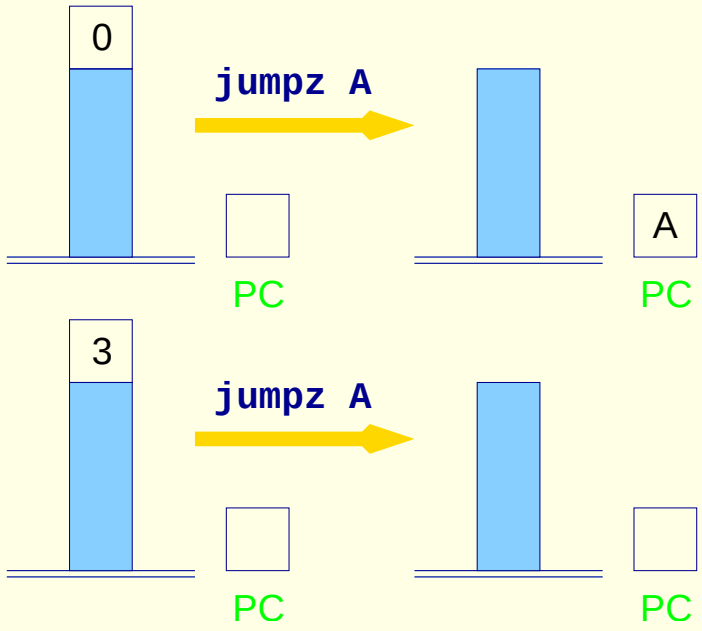
- Lihtsuse pärast kasutame hüpete sihtkohana sümbolmärgendeid, mis hiljem asendatakse absoluutsete aadressitega.
- Absoluutadressite asemel võib kasutada ka relatiivseid aadresseid; so. relatiivseid registri PC tegeliku väärtuse suhtes.
- Viimase lähenemise eeliseks on:
  - relatiivsed aadressid on reeglina *väiksemad*;
  - kood on *ringitõstetav* (ingl. relocatable).
- Käsk jump A teostab hüppe aadressile A; magasini ei muudeta.



PC = A;

Tingimuslikud laused ja tsüklid

- Käsk jumpz A teostab tingimusliku hüppe; eeldab magasini tipus ühte argumenti; kui argument on 0, siis toimub hüpe aadressile A, vastasel korral hüpet ei toimu.

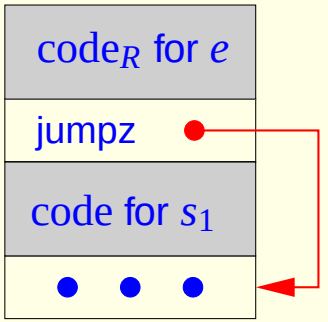



```
if (S[SP] ==0) PC = A;
SP−−;
```

Tingimuslikud laused ja tsüklid

- Üheharuline tingimuslause $s \equiv$ **if** $(e)\ s_1$
  - genereerime koodi tingimuse e ja lause s' jaoks;
  - lisame tingimusliku hüppekäsu nenede vahele.

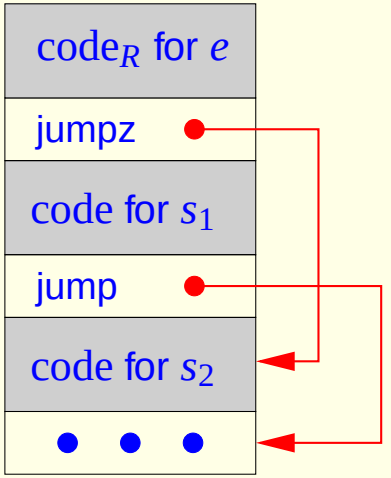
$$
\begin{array}{lll}
\text{code}\ (\textbf{if}\ (e)\ s_1)\ \rho & = & \text{code}_R\ e\ \rho \\
 & & \texttt{jumpz A} \\
 & & \text{code}\ s_1\ \rho \\
 & \text{A:} & \ldots
\end{array}
$$



Tingimuslikud laused ja tsüklid

- Kaheharuline tingimuslause $s \equiv$ **if** (e) s_1 **else** s_2

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



Tingimuslikud laused ja tsüklid

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

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

code s ρ emiteerib koodi:

```
loada 4          loada 4          A: loada 7
loada 7          loada 7             loada 4
ge               sub                 sub
jumpz A          storea 4            storea 7
                 pop                 pop
                 jump B           B: ...
```

(i) (ii) (iii)

Tingimuslikud laused ja tsüklid

- while-tsükkel $s \equiv$ **while** $(e)\ s_1$

code (**while** $(e)\ s_1$) ρ =

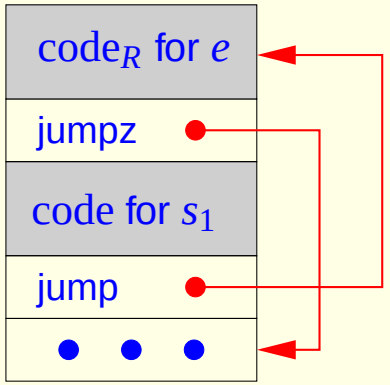
A: $\text{code}_R\ e\ \rho$

`jumpz B`

code $s_1\ \rho$

`jump A`

B: ...



Tingimuslikud laused ja tsüklid

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

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

code $s\ \rho$ emiteerib koodi:

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

(i) (ii) (iii)

Tingimuslikud laused ja tsüklid

- for-tsükkel $s \equiv$ **for** $(e_1;\ e_2;\ e_3)\ s_1$ on ekvivalentne while-tsükliga $e_1;$ **while** $(e_2)\ \{s_1\ e_3;\}$ (eeldusel, et s_1 ei sisalda continue-lauset)

$$
\begin{array}{lll}
\text{code}\ (\textbf{for}\ (e_1;\ e_2;\ e_3)\ s_1)\ \rho & = & \text{code}_R\ e_1\ \rho \\
 & & \texttt{pop} \\
 & \text{A:} & \text{code}_R\ e_2\ \rho \\
 & & \texttt{jumpz B} \\
 & & \text{code}\ s_1\ \rho \\
 & & \text{code}_R\ e_3\ \rho \\
 & & \texttt{pop} \\
 & & \texttt{jump A} \\
 & \text{B:} & \ldots
\end{array}
$$

Tingimuslikud laused ja tsüklid

- Mitmeharulised tingimuslaused tuleb reeglina transleerida mitmeteks üksteisesse sisestatud if-lauseteks.

switch (e) {		$x = e;$
case c_0 : ss_0 **break**;		**if** $(x == c_0)$ ss_0
case c_1 : ss_1 **break**;		**else if** $(x == c_1)$ ss_1
...	$\Longrightarrow$	...
case c_{k-1} : ss_{k-1} **break**;		**else if** $(x == c_{k-1})$ ss_{k-1}
default : ss_k		**else** ss_k
}		

- Kui märgendid sorteerida ja kasutada kahendotsimist, siis saab võrdluste arvu vähendada logaritmini märgendite arvust.

Tingimuslikud laused ja tsüklid

- Erijutudel võimalik konstantse ajaga hargnemine.
- Vaatame switch-lauset kujul:

$$
\begin{array}{lll}
s & \equiv & \textbf{switch}\ (e)\ \{ \\
 & & \quad \textbf{case}\ 0: \quad ss_0\ \textbf{break};\\
 & & \quad \textbf{case}\ 1: \quad ss_1\ \textbf{break};\\
 & & \quad \ldots \\
 & & \quad \textbf{case}\ k-1: \quad ss_{k-1}\ \textbf{break};\\
 & & \quad \textbf{default}: \quad ss_k \\
 & & \}
\end{array}
$$

Tingimuslikud laused ja tsüklid

$$
\begin{array}{lcllllll}
\text{code } s\ \rho & = & \text{code}_R\ e\ \rho & C_0: & \text{code } ss_0\ \rho & B: & \texttt{jump } C_0 \\
 & & \text{check } 0\ k\ B & & \texttt{jump D} & & \ldots \\
 & & & & \ldots & & \texttt{jump } C_k \\
 & & & C_k: & \text{code } ss_k\ \rho & D: & \ldots \\
 & & & & \texttt{jump D} & &
\end{array}
$$

- Makro check $0\ k\ B$ kontrollib, kas tingimusavaldise e R-väärtus on vahemikus $[0,k]$ ja seejärel teostab indekseeritud hüppe.
- ”Hüppetabeli” B i-s element sisaldab otsehüppe käsu i-ndale harule vastava koodi algaadressile.
- Iga haru lõpus on otsehüpe switch-lausest välja.

Tingimuslikud laused ja tsüklid

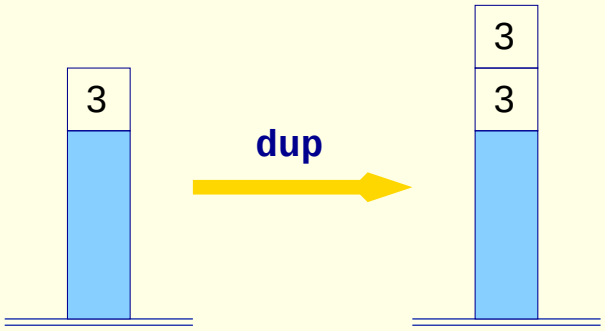
check 0 k B =

```
    dup         dup             jumpi B
    loadc 0     loadc k     A:  pop
    geq         le              loadc k
    jumpz A     jumpz A         jumpi B
```

- Tingimusavaldise e R-väärtust kasutatakse nii võrdluseteks, kui ka indekseerimiseks; seetõttu tuleb ta enne võrdlusi kopeerida.
- Kui R-väärtus ei ole vahemikus $[0, k]$, siis asendatakse ta enne hüpet konstandiga k.

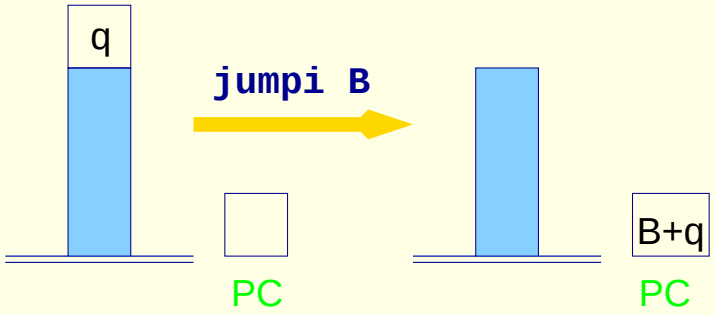
Tingimuslikud laused ja tsüklid

- Käsk `dup` kopeerib magasini ülemise elemendi



S[SP+1] = S[SP];
SP++;

- Käsk `jumpi B` teostab indekseeritud hüppe



PC = B + S[SP];
SP−−;

Tingimuslikud laused ja tsüklid

- ”Hüppetabeli” B võib paigutada kohe peale makrot check. See võimaldab kokku hoida mõned otsehüpped.
- Kui väärtuste vahemik algab u-st (ja mitte nullist), siis tuleb e R-väärtusest, enne tema kasutamist indeksina, lahutada u.
- Kui e kõik potentsiaalsel võimalikud väärtused on vahemikus $[0, k]$, siis pole makrot check vaja.

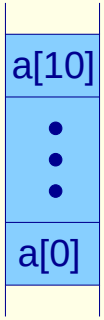
Massiivid, kirjed ning staatiline mäluhaldus

- Eesmärk: *staatiliselt* (so. kompileerimisajal) siduda iga muutujaga x fikseeritud (relatiivne) aadress $\rho\ x$.
- Eeldame, et baastüüpi muutujad (näit. **int**, ...) mahuvad ühte mälupessa.
- Seome muutujatega aadressid nende deklareerimise järjekorras alustades aadressist 1.
- Seega, deklaratsioonide $d \equiv t_1\ x_1; \ldots t_k\ x_k;$ (t_i on baastüüp) korral saame aadresskeskonna ρ sellise, et

$$\rho\ x_i = i, \qquad i = 1, \ldots, k$$

Massiivid, kirjed ning staatiline mäluhaldus

- Massiiv on staatiliselt määratud suurusega järjestikuste mälupesade hulk.
- Ligipääs kasutades täisarvulisi indekseid, mis algavad nullist.
- Massiivi aadress on tema esimese elemendi $a[0]$ aadress.
- Näide: deklaratsioon **int**$[11]a$; defineerib 11 elemendilise massiivi.



Massiivid, kirjed ning staatiline mäluhaldus

- Defineerime funktsiooni sizeof (tähistus $|\cdot|$), mis leiab tüübi mäluvajadused:

$$|t| = \begin{cases} 1 & \text{kui } t \text{ on baastüüp} \\ k \cdot |t'| & \text{kui } t \equiv t'[k] \end{cases}$$

- Seega, deklaratsioonide $d \equiv t_1\ x_1; \ldots t_k\ x_k;$ korral

$$\begin{aligned} \rho\ x_1 &= 1 \\ \rho\ x_i &= \rho_{i-1} + |t_{i-1}| \qquad i > 1 \end{aligned}$$

- Kuna $|\cdot|$ saab arvutada kompileerimise ajal, siis saab ka aadresskeskonna ρ arvutada kompileerimise ajal.

Massiivid, kirjed ning staatiline mäluhaldus

- Olgu $t\ a[c]$; massiivi a deklaratsioon.
- Siis i-nda komponendi algaadress on $\rho\ a + |t| \times (\text{rval of } i)$

$$\begin{aligned} \text{code}_L\ (a[e])\ \rho \quad = \quad & \texttt{loadc}\ (\rho\ a) \\ & \text{code}_R\ e\ \rho \\ & \texttt{loadc}\ |t| \\ & \texttt{mul} \\ & \texttt{add} \end{aligned}$$

- Üldjuhul võib massiiv olla esitatud avaldisena, mis tuleb enne indekseerimist väärtustada.
- C-s on deklareeritud massiiv *viitkonstant*, mille R-väärtus on massiivi algaadress.

Massiivid, kirjed ning staatiline mäluhaldus

$$
\begin{array}{lcll}
\text{code}_L\ (e_1[e_2])\ \rho & = & \text{code}_R\ e_1\ \rho & \\
 & & \text{code}_R\ e_2\ \rho & \\
 & & \texttt{loadc}\ |t| & \\
 & & \texttt{mul} & \\
 & & \texttt{add} & \\
\text{code}_R\ e\ \rho & = & \text{code}_L\ e\ \rho & e \text{ on massiiv}
\end{array}
$$

- NB! C-s on järgnevad (L-väärtustena) ekvivalentsed:

$$a[2] \qquad 2[a] \qquad a+2$$

- Normaliseerimine: massiivide nimed ja avaldised mis väärtustuvad massiiviks on enne indekseerimis-sulge, indeks-avaldis on sulgudes.

Massiivid, kirjed ning staatiline mäluhaldus

- Kirje on nime omavate, võimalik et erinevat tüüpi, komponentide hulk.
- Ligipääs komponentide nimede (selektorite) abil.
- Lihtsusena eeldame, et kirjekomponentide nimesid ei kasutata mujal.
  - Alternatiiv: hallata iga kirjetüübi *st* jaoks eraldi keskonda ρ_{st}.
- Kuulugu **struct** { **int** *a*; **int** *b*; } *x*; deklaratsioonide hulka:
  - kirje *x* relatiivne aadress on tema esimese pesa aadress;
  - komponentide aadressid on relatiivsed kirje algaadressi suhtes; so. ülaltoodud näites $a \mapsto 0,\ b \mapsto 1$.

Massiivid, kirjed ning staatiline mäluhaldus

- Olgu $t \equiv$ **struct** $\{\ t_1\ c_1; \ldots t_k\ c_k;\ \}$, siis

$$
\begin{array}{rcll}
|t| & = & \sum_{i=1}^{k} |t_i| & \\
\rho\ c_1 & = & 0 & \\
\rho\ c_k & = & \rho\ c_{i-1} + |t_{i-1}| & i > 1
\end{array}
$$

- Seega komponendi $x.c_i$ aadress on $\rho\ x + \rho\ c_i$.

$$
\begin{array}{rcl}
\text{code}_L\ (e.c)\ \rho & = & \text{code}_L\ e\ \rho \\
 & & \texttt{loadc}\ (\rho\ c) \\
 & & \texttt{add}
\end{array}
$$