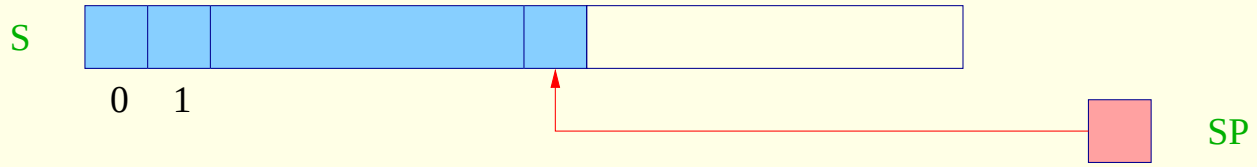
CMa — lihtsustatud C abstraktne masin

CMa arhitektuur

- Iga abstraktne masin määrab defineerib hulga *käske*, mis on täitmiseks abstraktsel riistvaral
- Seda abstraktset riistvara võib vaadelda kui teatavate andmestruktuuride kogumit, mida käsud kasutavad
- . . . ja mida juhib *täitmisaegne süsteem* (run-time system)

CMa arhitektuur

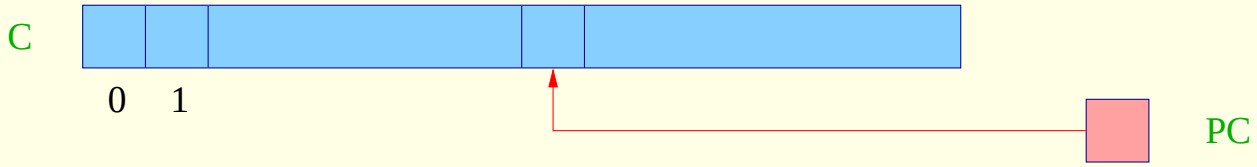
Magasin:



- S on mälupiirkond andmete hoidmiseks, kus uute elementide lisamine/eemaldamine käib LIFO printsiibil (Stack)
- SP on register, mis sisaldab ülemise (so. viimasena lisatud) elemendi aadressit (Stack Pointer)
- Lihtsustus: kõik mittestruktuurset tüüpi väärtused on sama suurusega ja mahuvad ühte magasini pessa.

CMa arhitektuur

Kood:



- C on mälupiirkond programmikoodi hoidmiseks, kus igas pesas on täpselt üks abstraktse masina käsk (Code)
- PC on register, mis sisaldab *järgmisena* täidetava käsu aadressit (Program Counter)
- Algselt PC sisaldab aadressit 0; so. C[0] sisaldab kõige esimesena täidetavat käsku

CMa arhitektuur

Programmi täitmine:

- Masin laadib käsu C[PC] registrisse IR (Instruction-Register), seejärel suurendab registrit PC ühe võrra ning lõpuks täidab selle käsu:

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

- Käsu täitmine võib muuta registri PC sisu (hüpped)
- Masina peatsükli katkestab käsk halt, tagastab kontrolli väliskeskkonnale
- Ülejäänud käsud toome sisse järk-järgult vastavalt vajadusele

Lihtsad avaldised ja omistamine

Ülesanne: väärtustada avaldis $(1+7)*3$

See tähendab: genereerida käskude jada, mis

- leiab avaldise väärtuse ja
- lisab selle magasini tippu.

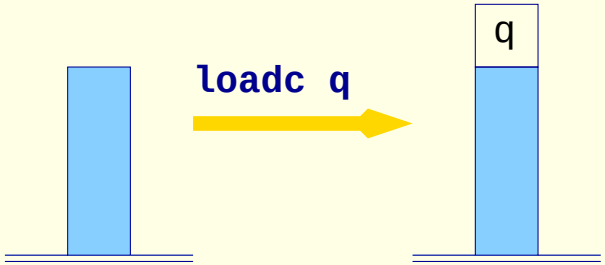
Idee:

- kõigepealt arvutame alamavaldiste väärtused,
- salvestame need magasini tippu ning
- seejärel rakendame vastavat operaatorit.

Lihtsad avaldised ja omistamine

Üldprintsiip:

- käsud eeldavad, et nende argumendid on magasini ülemised elemendid,
- käsu täitmine tarvitab oma argumendid ära,
- tulemus salvestatakse magasini tippu.

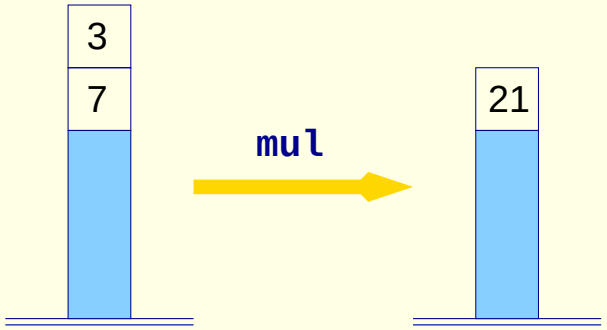


SP++;
S[SP] = q;

Käsk `loadc q` ei oma argumente magasinis ja salvestab konstandi `q` magasini tippu.

NB! Registri SP sisu on joonisel esitatud ainult kaudselt magasini kõrguse kaudu.

Lihtsad avaldised ja omistamine



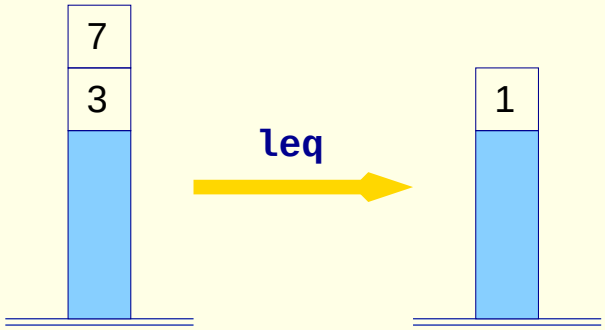

```
SP−−;
S[SP] = S[SP] * S[SP+1];
```

Käsk `mul` eeldab magasinis kahte argumenti, tarvitab need ära ja salvestab nende korrutise magasini tippu.

Teised binaarsed aritmeetilised ja loogilised käsud ning loogilised operaatorid `add`, `sub`, `div`, `mod`, `and`, `or`, `xor`, `eq`, `neq`, `le`, `leq`, `ge` ja `geq` töötavad analoogselt.

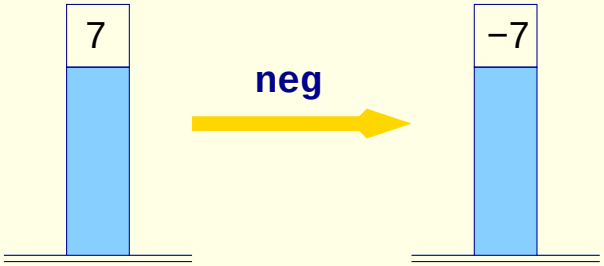
Lihtsad avaldised ja omistamine

Näide: Operaator `leq`



NB! 0 esitab väära väärtust, kõik teised täisarvud on tõesed väärtused.

Unaarsed operaatorid `neg` ja `not` tarvitavad ühe argumendi ja produtseerivad ühe tulemuse.



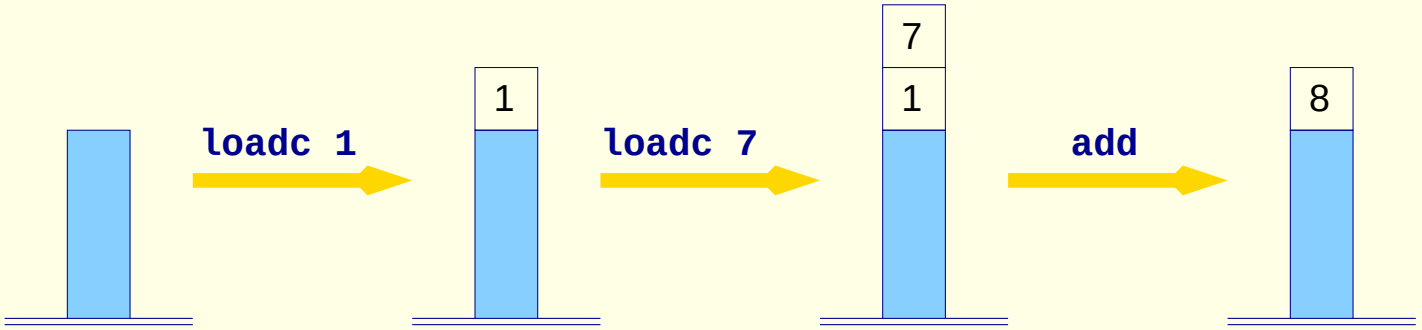

```
S[SP] = −S[SP];
```

Lihtsad avaldised ja omistamine

Näide: Avaldisele $1 + 7$ vastab käskude jada

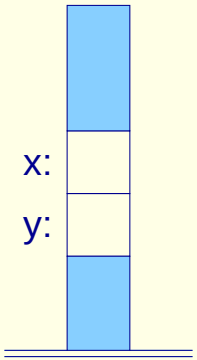
```
loadc 1
loadc 7
add
```

Selle koodi täitmine annab:



Lihtsad avaldised ja omistamine

- Muutujatele vastavad magasini S pesad:



- Koodi genereerimist kirjeldavad funktsioonid code, code_L ja code_R.
- Argumendid: transleeritav süntaktiline konstruktsioon *e* ja funktsioon ρ (aadresskeskond), mis igale muutujale *x* seab vastavusse tema relatiivse aadressi magasinis.

Lihtsad avaldised ja omistamine

- Muutujaid kasutatakse kahel erineval moel.
- Näiteks omistuslauses $x = y + 1$ oleme huvitatud muutuja y *väärtusest*, kuid muutuja x *aadressist*.
- Muutuja süntaktiline asukoht määrab, kas me vajame tema *L-väärtus* või *R-väärtust*

muutuja L-väärtus = tema aadress

muutuja R-väärtus = tema väärtus

$\text{code}_L\ e\ \rho$	emiteerib keskonnas ρ avaldise e L-väärtust arvutava koodi
$\text{code}_R\ e\ \rho$	sama R-väärtuse jaoks

- NB! Mitte igal avaldisel pole L-väärtust (näit.: $x + 1$)

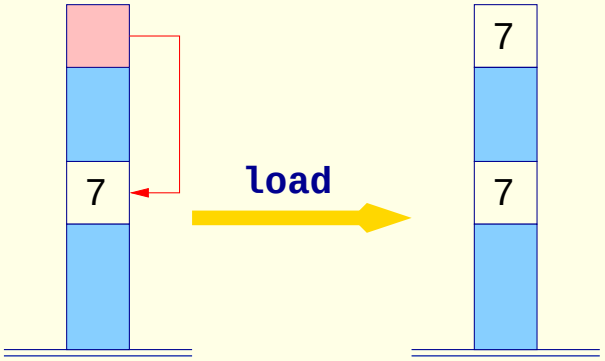
Lihtsad avaldised ja omistamine

$$
\begin{array}{lcl}
\text{code}_R\ (e_1 + e_2)\ \rho & = & \text{code}_R\ e_1\ \rho \\
 & & \text{code}_R\ e_2\ \rho \\
 & & \texttt{add} \\
 & \ldots & \text{analoogselt teiste binaarsete operaatorite jaoks} \\
\text{code}_R\ (\text{-}e)\ \rho & = & \text{code}_R\ e\ \rho \\
 & & \texttt{neg} \\
 & \ldots & \text{analoogselt teiste unaarsete operaatorite jaoks} \\
\text{code}_R\ q\ \rho & = & \texttt{loadc}\ q \\
\text{code}_L\ x\ \rho & = & \texttt{loadc}\ (\rho\ x) \\
 & \ldots &
\end{array}
$$

Lihtsad avaldised ja omistamine

$$\text{code}_R\ x\ \rho \quad = \quad \text{code}_L\ x\ \rho$$
$$\texttt{load}$$

Käsk `load` salvestab magasini tippu selle magasini pesa sisu, mille aadress on ülemises pesas

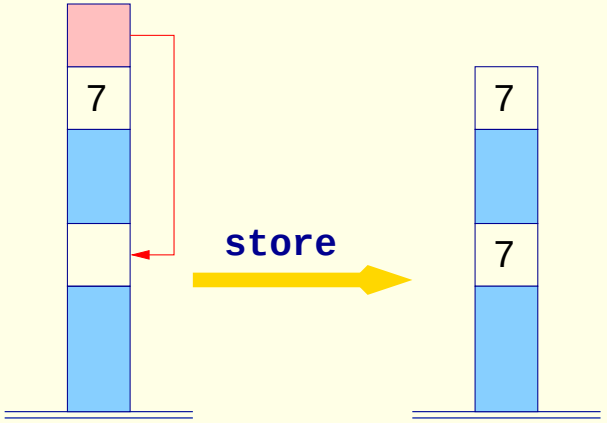


S[SP] = S[S[SP]];

Lihtsad avaldised ja omistamine

$$\begin{aligned} \text{code}_R\ (x = e)\ \rho \quad &= \quad \text{code}_R\ e\ \rho \\ & \qquad\quad \text{code}_L\ x\ \rho \\ & \qquad\quad \texttt{store} \end{aligned}$$

Käsk `store` kirjutab ülalt teise pesa sisu pessa, kuhu viitab magasini ülemine pesa ja jätab kirjutatud väärtuse ka magasini tippu.



S[S[SP]] = S[SP−1];
SP−−;

NB! Erineb Wilhelm/Maurer'i raamatus toodud vastavast P-machine'i käsust

Lihtsad avaldised ja omistamine

Näide: Olgu $e \equiv x = y - 1$ ja $\rho = \{x \mapsto 4, y \mapsto 7\}$, siis $\text{code}_R\ e\ \rho$ emiteerib koodi:

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

Täiustusi: sagedasti esinevate käskude kombinatsioonide jaoks võib sisse tuua uusi käske, näiteks

```
loada q   =  loadc q
             load

storea q  =  loadc q
             store
```