SPE@

Mis on SPE@?

- SPE@(**S**imple **P**rocessor **E**mulator and **T**ranslator)
- Lihtsa protsessori emulaator
- Selle protsessori masinkoodile orienteeritud assembleri translaator
- Vahend mugavalt masinkoodi ja/või assembleri õppida

Protsessori Emulaator

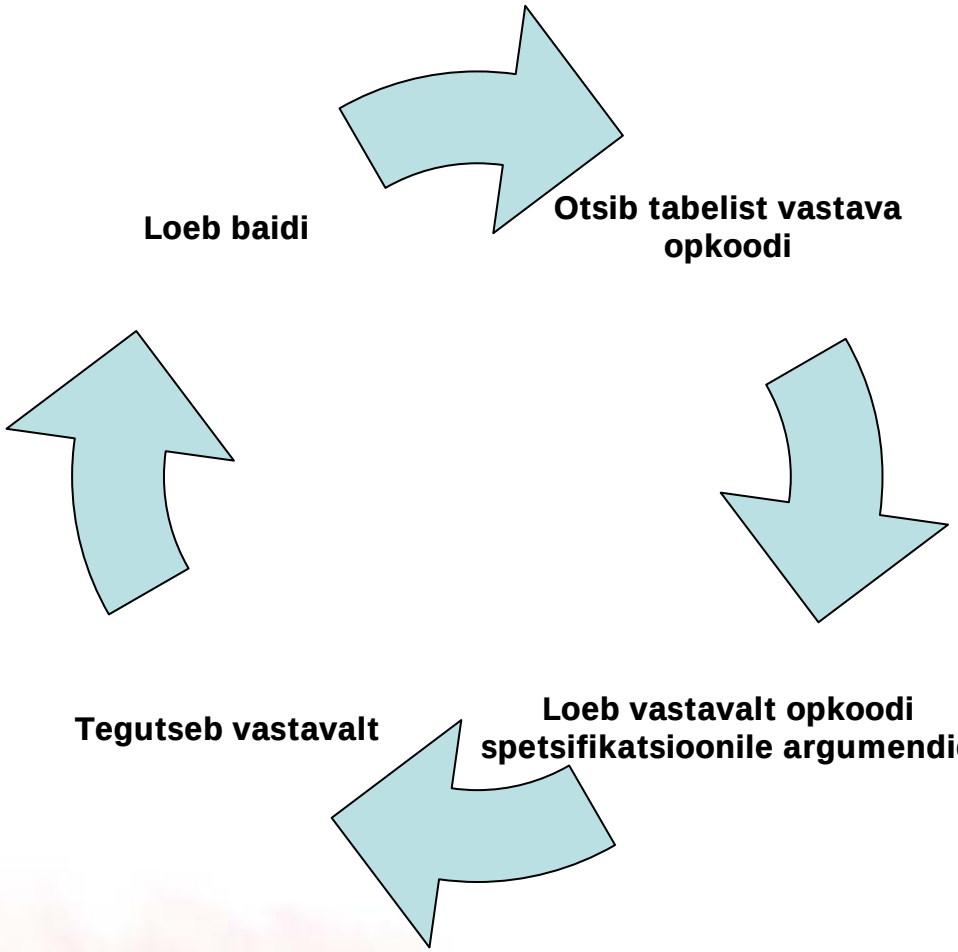
- Baitmasin, mälu on piiratud kasutaja arvuti ressursidega
- 15 registrit + virtuaalne register, summaator – 32 bitised
- 100 opkoodi
- Sisenend ja Väljund

Protsessori Emulaator

Kuna opkoodi suuruseks on 1 bait siis saavad olla võimalikud opkoodid vahemikus 00..ff ehk saab olla 256 erinevat opkoodi

- Probleem: Kui oleks vaja rohkem kui 256 opkoodi?
- Lahendus: Grupeerime opkoodid lehtedele. 00..09 on lehe määrangust sõltumatud 10 koodi, 10..ff on lehest sõltuvad 246 opkoodi. Lehel 0 on standard protsessori käsud, Lehel 1 mäluoperatsioonid jne.
- Järeldus: Protsessor on lõpmatuseni laiendatav(realisatsioonis $2^{64} \times 246$ opkoodi)

Protsessori Emulaator



Translaator

- Käib lause lauselt koodi läbi ja tekitab vastavalt masinkoodi
- Toetab muutujate kasutust ja pointereid nendele
- Ei kontrolli süntaksit ega optimeeri

Assembler

- Ei kontrolli süntaksit ega optimeeri
- Toetab muutujate kasutust ja pointereid nendele
- Arvud kas kümnend süsteemis (255,165) või kuueteistkümnend süsteemis (ffh,a5h)
- Registreid märgitakse $(1..f), $0 tähistab sumaatorit

Assembler

add 15**h** **:** liidab summaatorile arvu 21
set ah **:** määrab summatoriks 10da registri
add 3 **:** liidab sumaatorile arvu 3
set 3 **:** määrab sumaatoriks 3da registri
add **$**1,**$**ah **:** omistab sumaatorile 1. ja 10da registri summa
out **$**0 **:** väljastab summatori
stop **:** lõpetab töö

1115f00a1103f00312010af40000

Assembler

- Rida arvatakse *Jumplabel*'iks kui ta algab ":" sümboliga
- Andmeid deklareeritakse direktiiviga "dd" (dd 5)
- Muutujate deklareerimine nagu andmetegi, lisama peab ainult nime (dd viis,5)
- Mälu aadresse tähistatakse "&" sümboliga (&9), muutuja aadressi "[<muutuja nimi>]" ([muutuja1])

Assembler

out n : väljastab väärtuse kohal n
out m : väljastab väärtuse kohal m
out [n] : väljastab aadressi n
out [m] : väljastab aadressi m
stop : lõpetab töö
dd n,12h : määrab n'ile aadressi 9
dd m,13h : määrab m'ile aadressi a

f509f50af309f30a001213

Näite Programm #1

```
in $1          : määrab 1. registrile sisendi väärtuse
:algus         : jumplabel
set 1          : määrab summaatoriks 1. registri
dec            : lahtutab sumaatori väärtusest ühe
in $3          : määrab 3dale registrile sisendi väärtuse
set 4          : määrab summaatoriks 4da registri
add $4,$3      : omistab sumaatorile 4da ja 3da registri summa
set 2          : määrab summaatoriks 2. registri
cmp $1,0       : võrdleb 1. registrit nulliga
jin :lõpp,:algus: kui võrdus kehtib siis liigub :lõpp, kui ei siis :algus
:lõpp          : jumplabel
out $4         : väljastab tulemuse
stop           : lõpetab programmi töö
```

f701f00126f703f004120403f002a00100931402f40400

Näite Programm #2

```
:algus          : jumplabel
set $15         : määrab summaatoriks 15da registri väärtuse
inc             : liidab sumaatori väärtusesele ühe
set 15          : määrab summaatoriks 15da registri
inc             : liidab sumaatori väärtusesele ühe
set 14          : määrab summaatoriks 14da registri
cmp $15,13      : võrdleb 1. registrit nulliga
jin :null,:start : kui võrdus kehtib siis liigub :null, kui ei siis :algus
:null           : jumplabel
mov $15,0       : omistab 15dale registrile väärtuse 0
jmp :algus      : liigub labelile :algus
stop            : lõpetab programmi töö
```

f10f16f00f16f00ea00f0d930e00770f00900000

PEPMO@

PEPMO@ on SPE@'i lihtsustatud järglane, ning kuna on kirjutatud Javas siis jookseb "igal" masinal.
Täpsem informatsioon
http://www.root.ee/pepmo

SPE@ ja PEPMO@ assembleri ülevaade asub aadressil:
http://www.root.ee/pemo/manual.html