Geheue-organisasie

- Ken geheue toe tydens looptyd vir elke aktivering van 'n prosedure (aktiveringsrekord).
- Stapel is effektiewe tegniek.
- Gebruik indeksregisters om huidige aktiveringsrekord af te baken:
  - SP (R30) wys na top van stapel
  - FP (R29) wys na onderkant van aktiveringsrekord
- Stoor inligting in aktiveringsrekord om terugkeer na vorige aktiveringsrekord te ondersteun(terugkeeradres, vorige FP).

Lokale veranderlikes, parameters

- Ken geheue toe vir plaaslike veranderlikes.
- Parameters word vooraf op stapel gedruk en verwyder tydens terugkeer.

Begin van prosedure:

```
PSH   LNK,SP,4    push link
PSH   FP,SP,4     push FP
ADD   FP,0,SP     FP := SP
SUBI  SP,SP,n     space for local variables
```

'

Einde van prosedure:

```
ADD   SP,0,FP     SP := FP
POP   FP,SP,4     pop FP
POP   LNK,SP,4    pop link
RET   0,0,LNK     return jump
```


Prosedure roepe

- Aktiveringsrekords word aanmekaar geskakel in die volgorde wat die prosedures geroep is (dinamiese skakel—FP)
- CISC verwerkers het spesiale instruksies om prosedure roepe te ondersteun.
- Sulke instruksies spaar 'n paar grepe geheue en miskien 'n paar mikrosekondes uitvoertyd.
- Tipiese instruksies:
  - enter n (save fp and allocate n bytes)
  - exit (restore previous fp)
  - return n (jump back and remove parameters)

Lokale veranderlikes

- Adres van lokale verandelikes is relatief tot die basis van die aktiveringsrekord.
- Hierdie basis-adres word gestoor in FP.
- Adresse toegeken in volgorde soos veranderlikes verklaar is.
- Twee 32-bis woorde op stapel (vorige FP, terugkeeradres)
- Laaste veranderlike is by FP+8
- Lokale veranderlikes het negatiewe verplasings vanaf FP.
- Per verwysing parameters se *adres* is op die stapel.

Globale veranderlikes

- Lokale veranderlikes is by vlak 0.
- Vir elke nestingsvlak tel een by die vlak van 'n veranderlike.
- Nodig om terugwaarts te soek in stapel vir globale veranderlikes deur 'n *statiese* skakel te volg—ekstra inligting gestoor by die basis van elke aktiveringsrekord.
- Meer effektiewe implementering is moontlik deur slegs *twee* vlakke toe te laat: globaal en lokaal.


```
PROCEDURE P(x: INTEGER; VAR y: INTEGER);
BEGIN x := y; y := x; P(x,y) END P
```

```
   PSH   LNK,SP,4
   PSH   FP,SP,4
   ADD   FP,0,SP
   SUBI  SP,SP,0      no local variables
   LDW   1,FP,8
   LDW   2,1,0
   STW   2,FP,12      x := y
   LDW   1,FP,8
   LDW   2,FP,12
   STW   2,1,0        y := x
   LDW   1,FP,12      x
   PSH   1,SP,4
   LDW   1,FP,8       adr(y)
   PSH   1,SP,4
   BSR   0,0-14       P(x,y)
   ADD   SP,0,FP
   POP   FP,SP,4
   POP   LNK,SP,12    pop link and parameters
   RET   0,0,31
```


Standaard prosedures

- Prosedures wat nie verklaar hoef te word nie (ABS, INC, DEC).
- Kode word direk genereer sonder BSR instruksie (effektief).
- Oberon-0: Read, Write, WriteHex, WriteLn (ooreenstemmende masjieninstruksies).
- Plaas inskrywings vir elke prosedure in simbooltabel tydens inisialisering.

```
Read(x); Write(x); WriteLn
```

```
   READ   1,0,0
   STW    1,0,-4      x
   LDW    1,0,-4      x
   WRD    0,0,1
   WRL    0,0,0
```


Funksies

- Prosedures wat aktiveer word as deel van uitdrukkings.
- Resultaat word gewoonlik teruggestuur in 'n register (effektief).
- Resultaat se tipe moet dus in 'n register pas en komplekse tipes is dus ontoelaatbaar.
- Indien die resultaat meer kompleks moet wees, moet dit via die stapel teruggestuur word deur ruimte te reserveer volgens die tipe.