3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von **dynamischen Feldern**:

var feld: **array**[u1..o1,…,uk..ok] **of integer**;

//ui, oi nicht alle konstant; z.B. Parameter

Speicherbelegung: (Felddeskriptor)

0	fiktive Anfangsadresse
1	Feldgröße
2	Subtr. für fik. Anf.Adresse
3	u1
4	o1
…	…
2k+1	uk
2k+2	ok
2k+3	d2
…	…
3k+1	dk

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von dynamischen Feldern:

Befehl	Bedeutung	Beding.	Ergebn.
ldd *q*	SP := SP+1; STORE[SP] := STORE[STORE[SP -3] +q]	(a, *T1, T2*)	(*a, T1, T2,i*)
dpl *T*	SP:= SP+1; STORE[SP] := STORE[SP-1]	(*T*)	(*T,T*)
sli T2	STORE[SP-1] := STORE[SP]; SP := SP-1	(*T1,T2)*	(*T2*)

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von dynamischen Feldern:

```
codeLd b[i1,..,ik] p = ldc a p(b);   //Deskriptoradresse
                       codeId [i1,..,ik] g p
```

mit

```
codeId  [i1,..,ik] g p = dpl a ;        //dupliziert obersten Kellereintrag
                         ind a;         //ersetzt Kopie durch fiktive AnfAdr.
                         ldc i 0;
                         codeR i1 p; add i; ldd  2k+3; mul i;
                         codeR i2 p; add i; ldd  2k+4; mul i;
                              …
                         codeR ik-1 p; add i; ldd 3k+2; mul i;
                         codeR ik; add i;
                         ixa g;
                         sli a;         //schiebt STORE[SP] auf STORE[SP-1]
```

3. Sprachkonzepte und ihre Übersetzungen

Übungsaufgabe: Gegeben sei folgende Prozedeur:

```
proc p;
    var b: array[u1..o1,u2..o2] of integer;
    i,j: integer;
    begin
      …
      b[i,j] := b[i-1,j+1]+1;
    end;
```

1. Bei Eintritt in die Prozedur haben die globalen Variablen folgende Werte: u1 = -4, o1 = 5, u2 = 1, o2 = 10. Es sei p(b)=5 die Adresse des Felddeskriptors, Adresse von b[u1,u2] sei 30. Geben Sie den Felddeskriptor für b an.
2. Die Variable i erhält den Wert 2, j den Wert 4. Übersetzen Sie den linken Teil der Zuweisung im Rumpf von p und zeigen Sie graphisch den (aktuellen) Keller pro P-Befehl der Übersetzung.

   An welcher Adresse befindet sich b[i,j]?

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von **Verbunden**:

```
var   i,j: integer;
      v: record
           a:integer;
           b: bool
         end;
```

Speicherbelegung:

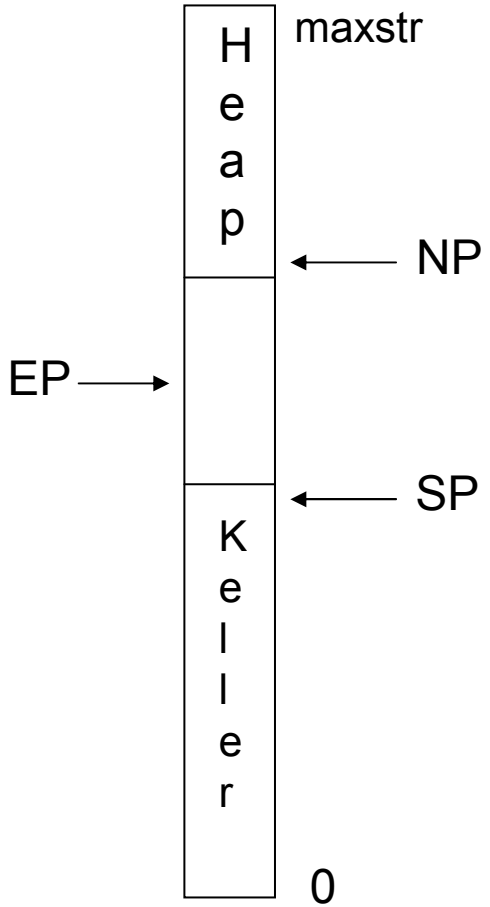
- Verbundvariable v: erste freie Speicherzelle
- Komponenten a, b: Relativadressen innerhalb von v
  p(a) = 0, p(b) = 1
- im Beispiel p(i)=5, p(j)=6, p(v)=7, p(a)=0, p(b)=1
- p(ci) = ∑ gr(Tj) für j=1 bis i-1 //allg.: Relativadr. einer Komponente
- gr(v) = ∑ gr(Ti) für i=1 bis n //Größe von v bei n Komponenten
- Adressieren von ci in v: ldc a p(v); inc a p(ci);

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von **Zeigern / dyn. Speicherbelegung**:

Eigenschaften:

- kellerartiger Speicher passt nicht
- -> eigener Speicherbereich: Heap
- Heap wächst in Richtung Keller
- Register NP, Hilfsregister EP
- P-Befehl new



3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von **Zeigern / dyn. Speicherbelegung**:

P-Befehl new:

Befehl	Bedeutung	Beding.
new	if NP-STORE[SP] <= EP then error(„store overflow“) else NP.=NP – STORE[SP]; STORE[STORE[SP-1]]:=NP; SP := SP – 2 fi	(a,i)

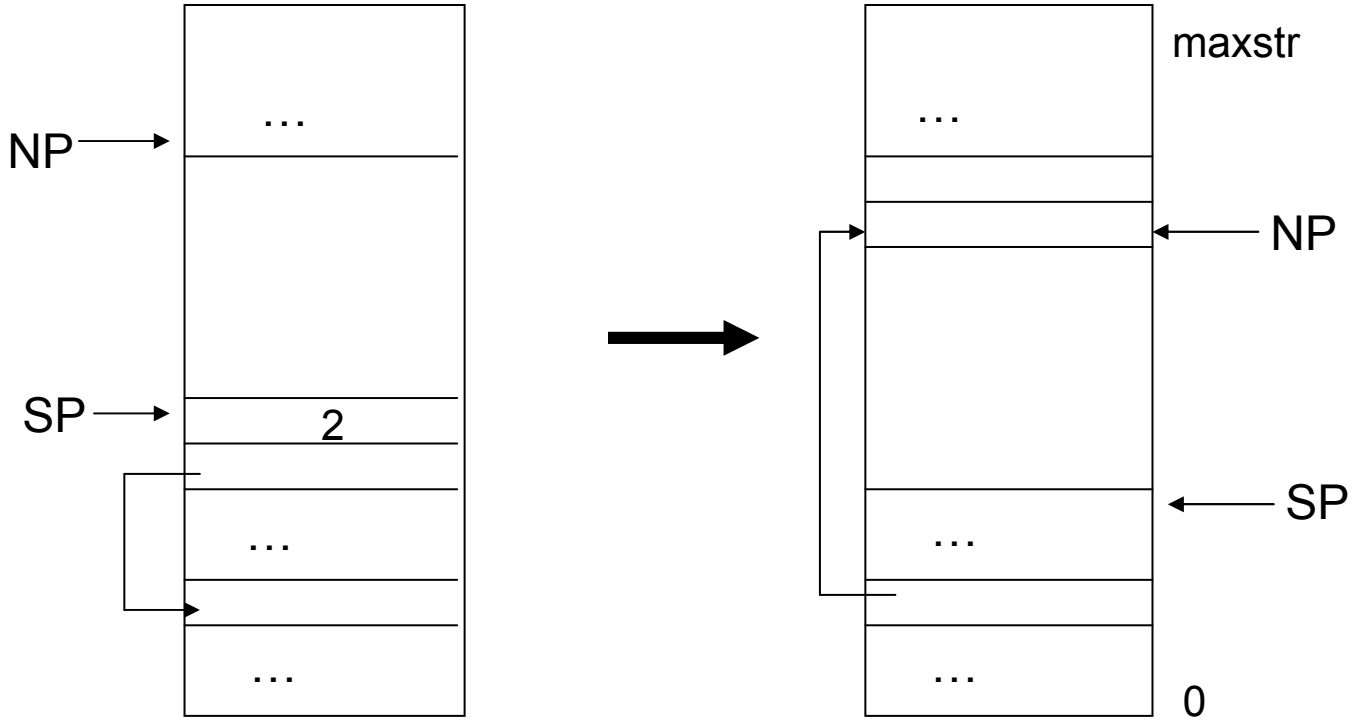
in STORE[SP]: Größe des zu erzeugenden Objekts

in STORE[SP-1]: Adresse des Zeigers auf Objekt

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung new-Anweisung:

code(new(x)) p = ldc a p(x); ldc i gr(T); new



3. Sprachkonzepte und ihre Übersetzungen

Dereferenzierung mittels ind :

Beispiel:

```
type t=record
          a:array[-5..5,1..9] of integer;
          b: ^t
        end;
var     i,j:integer;        //p(i)=5, p(j)=6
        pt: ^t;             //p(pt)=7
```

code(pt^.b^.a[i,j]) = ldc a 7; ind a; inc a 99; ind a; inc a 0;codeI[i, j] 1 p;

3. Sprachkonzepte und ihre Übersetzungen

Übersetzung von **Funktionen/Prozeduren**:

Eigenschaften:

- Deklaration: Name, formale Parameter, lokale Deklarationen, Rumpf, bei Funktionen: plus Ergebnistyp
- Aufrufbaum: Kette von Prozeduraufrufen, Wurzel: Hauptprogramm
- Inkarnationswege von P: Wege von Wurzel bis Aufruf von P
- lokale Namen in P: formale Parameter, lokale Deklarationen
- bei Prozeduraufruf Speicher für form. Parameter, lokale Deklarationen, Zwischenergebnisse
- Freigabe bei Prozedur-Ende
- globale Namen, Zugriff über statische o. dynamische Bindung

3. Sprachkonzepte und ihre Übersetzungen

Beispiel statische vs. dynamische Bindung:

	statisch:	dynamisch:

var x;
procedure p
  procedure q
  var x;
  p;
	x in p: äußere Dekl.	x in p: Dekl. in q

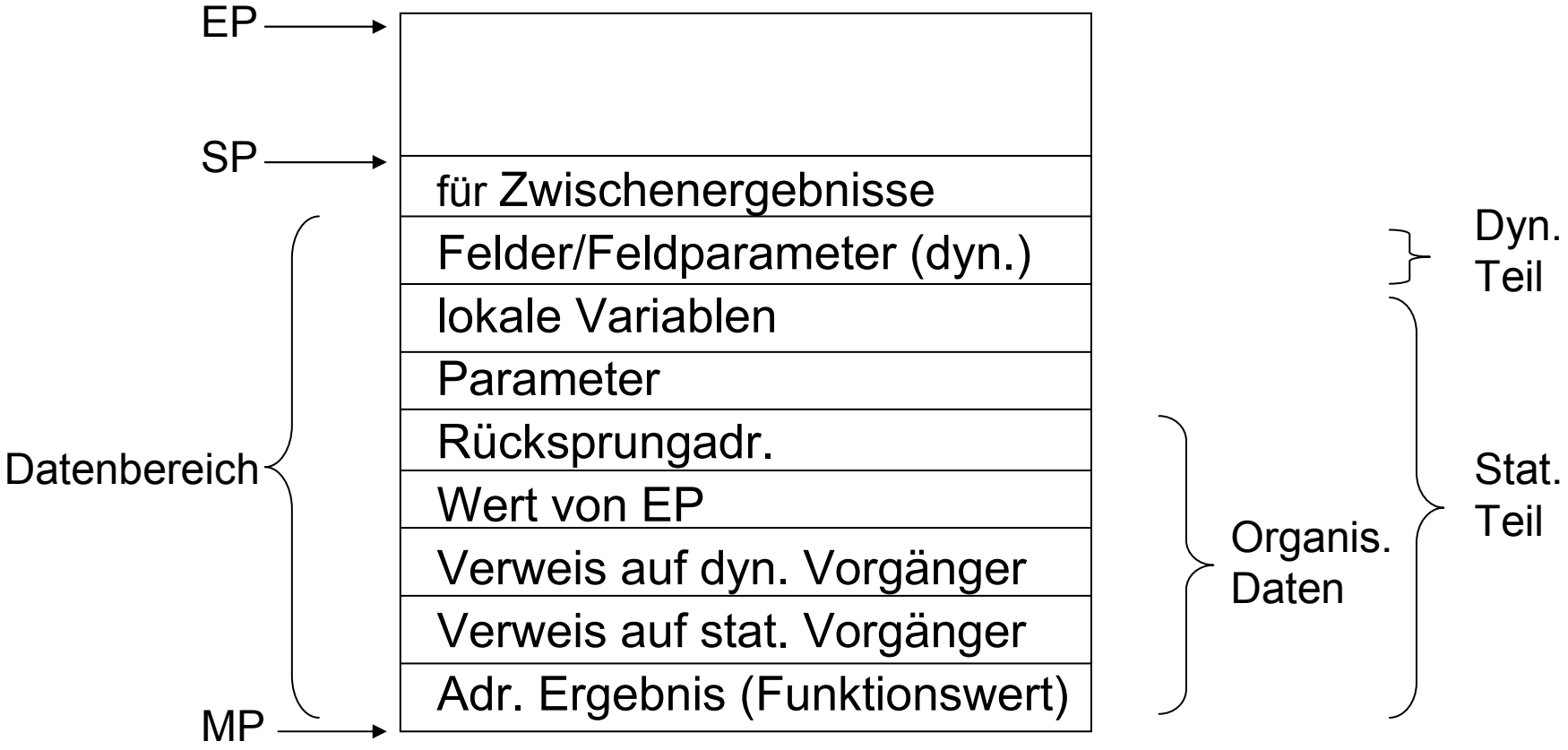
end; x; q; end; p;	x in p: äußere Dekl.	x in p: äußere Dekl.

Im Folgenden statische Bindung

3. Sprachkonzepte und ihre Übersetzungen

STORE bei Funktionen/Prozeduren:

pro Aufruf Kellerrahmen /Activation record



Zugriff auf Daten: (statische) Relativadresse zu MP

3. Sprachkonzepte und ihre Übersetzungen

Zugriff auf Kellerdaten bei Funktionen/Prozeduren:

- über (statische) Relativadresse zu MP
- bei nichtlokalen Namen Verweis auf statischen Vorgänger entsprechend der Differenz der Schachtelungstiefe oft verfolgen
- → weitere P-Befehle zum Laden und Speichern von Werten bzw. Adressen bei gegebener Differenz der Schachtelungstiefen und einer Relativadresse innerhalb des activation records
- Weitere P-Befehle (u.a.):

  mst mark stack, legt organisatorische Daten auf Stack

  ssp set stack pointer (SP:=MP+p-1), p Größe stati. Teil

  sep set extreme stack pointer

  cup call user procedure, Sprung zur Anfangsadr. der neuen Prozedur, vorher Retten der Rückspr.adr.

  retf, retp Rücksprung aus Funktion/Prozedur

3. Sprachkonzepte und ihre Übersetzungen

Fragestellungen zu Kapitel 3:

- Wie ist ein Datenspeicher aufgebaut?
- Welche besonderen Register haben Maschinen?
- Welche Maschinen(Assembler)befehle benötigt man zum Laden und Speichern von Werten?
- Übersetzung einfacher Ausdrücke/Anweisungen in P-Befehle.
- Welche Sprungbefehle gibt es?
- Wie werden statische, wie dynamische Felder übersetzt (Prinzip)?
- Was ist der Heap und welche Aufgabe hat er?
- Welche Daten müssen bei Funktionen/Prozeduren auf dem Datenspeicher abgelegt werden?