

Metody přidělování paměti

Základní způsoby: -Statické (přidělení paměti v čase překladu)
-Dynamické (přiděleno v run time) v zásobníku
na haldě

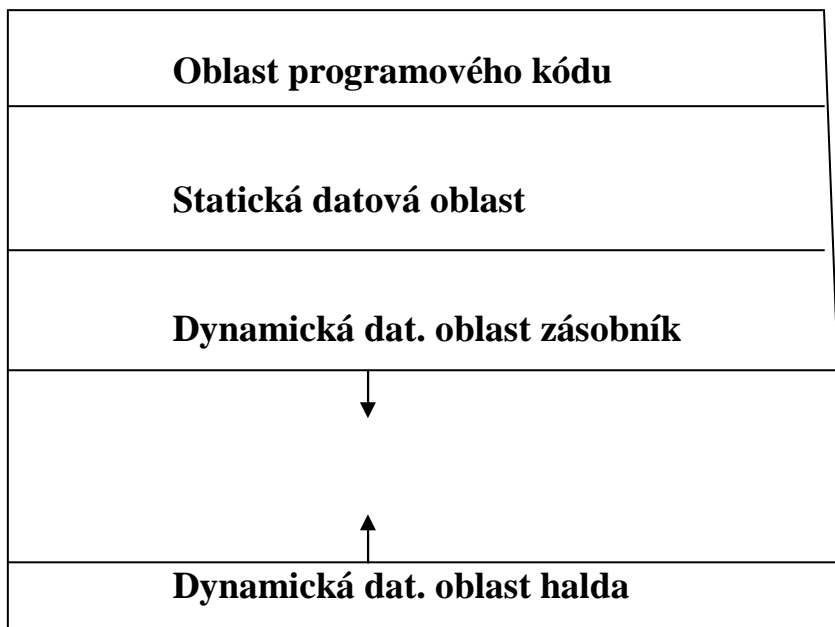
Důležitá hlediska jazykových konstrukcí:

- Dynamické typy
- Dynamické proměnné
- Rekurze
- Konstrukce pro paralelní výpočty

Podstatný je rovněž způsob:

- Omezování existence entit v programu (namespace, package, blok...)
- Určování přístupu k nelokálním entitám
na základě statického vnořování rozsahových jednotek,
na základě dynamického vnoření rozsahových jednotek.

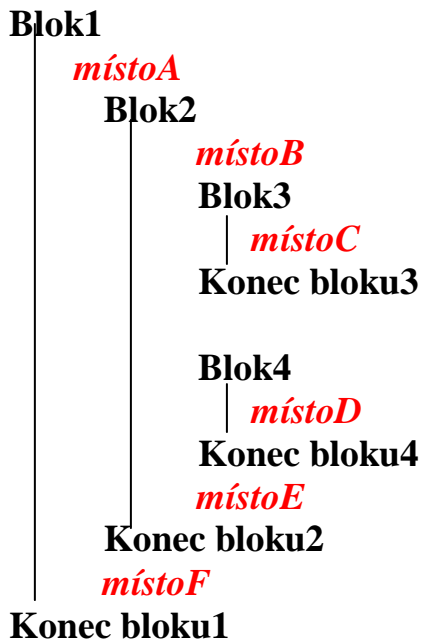
Rozdělení paměti cílového programu



Statické přidělování paměti lze použít pro:

- Globální proměnné
- Static proměnné
- Proměnné jazyka bez rekurze (i s vnořenou blokovou strukturou)

Př.



Statické přidělování lze realizovat pomocí zásobníku

Ukažme obsah zásobníku v různých okamžicích výpočtu

Bloky

		3	4		
	2	2	2	2	
1	1	1	1	1	1
Místo A	B	C	D	E	F

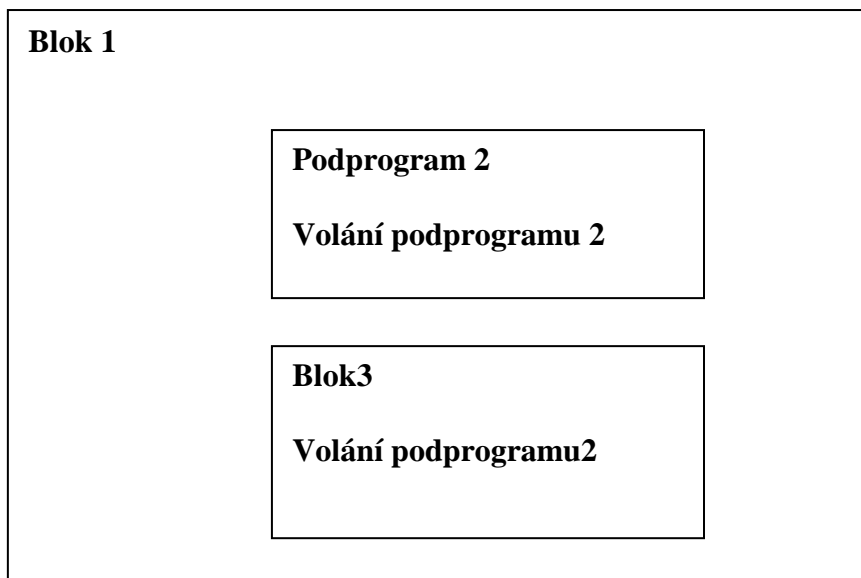
Vnořování podprogramů (funkcí, metod) je ale složitější, viz dále.

Dynamické přidělování v zásobníku

Část paměti přidělovaná při vstupu výpočtu do rozsahové jednotky programu se nazývá **Aktivační Záznam** (AZ představuje lokální prostředí výpočtu). Obsahuje místo pro:

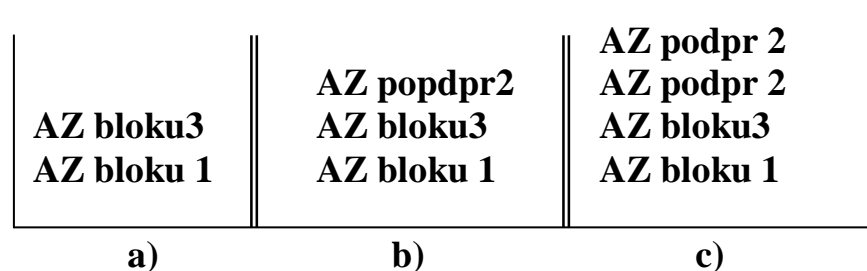
- Lokální proměnné
- Parametry (je-li rozsahovou jednotkou podprogram či funkce)
- Návratovou adresu („ „)
- Funkční hodnotu (je-li rozsahová jednotka funkcí)
- Pomocné proměnné pro mezivýsledky (také možno v registrech)
- Další informace potřebné k uspořádání aktivačních záznamů

Př.1

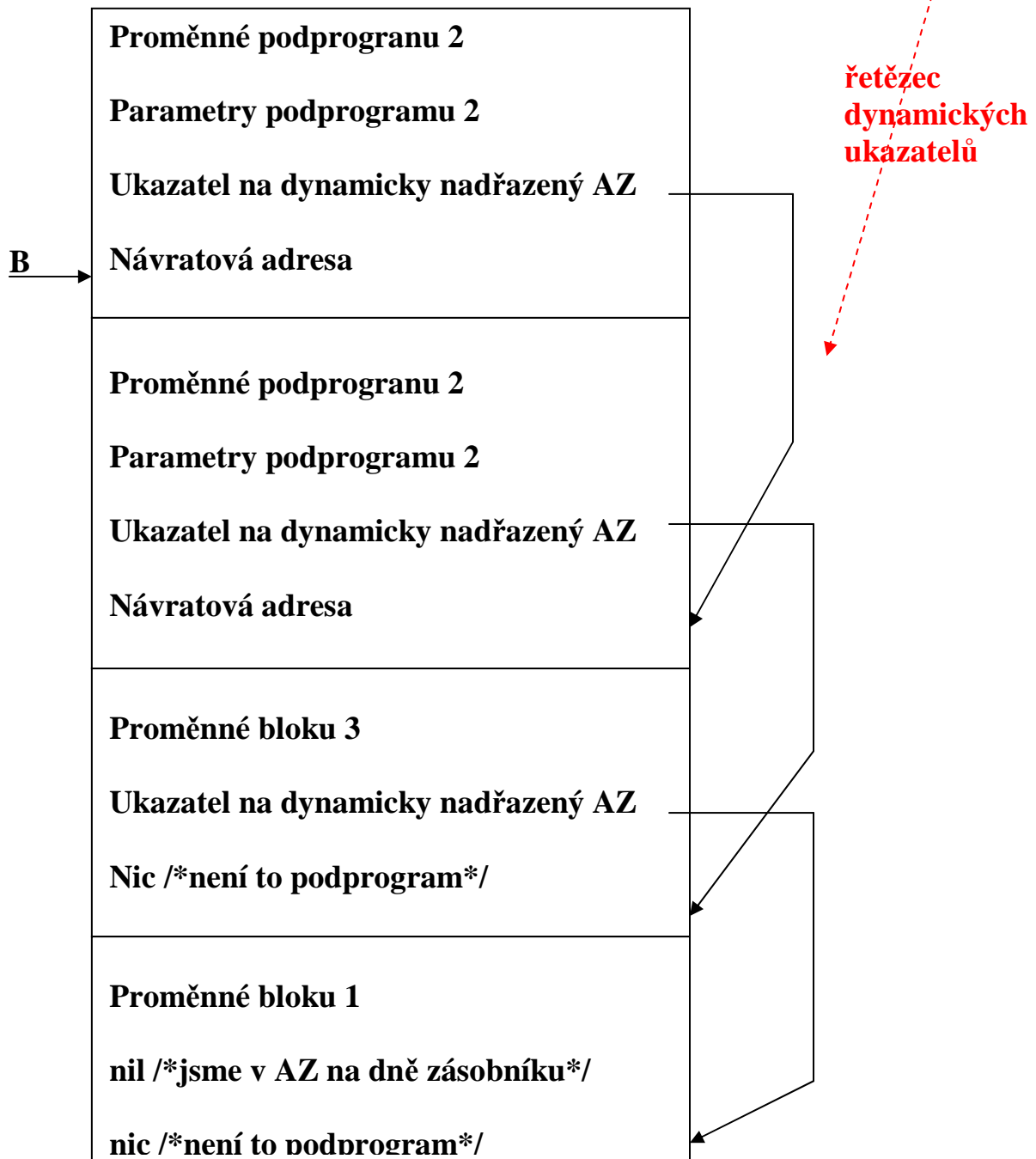


? stav výpočtového zásobníku v různých časech výpočtu

- a) Při vstupu do bloku 3
- b) Při prvním volání podprogramu 2
- c) Při rekurzivním vyvolání podprogramu 2



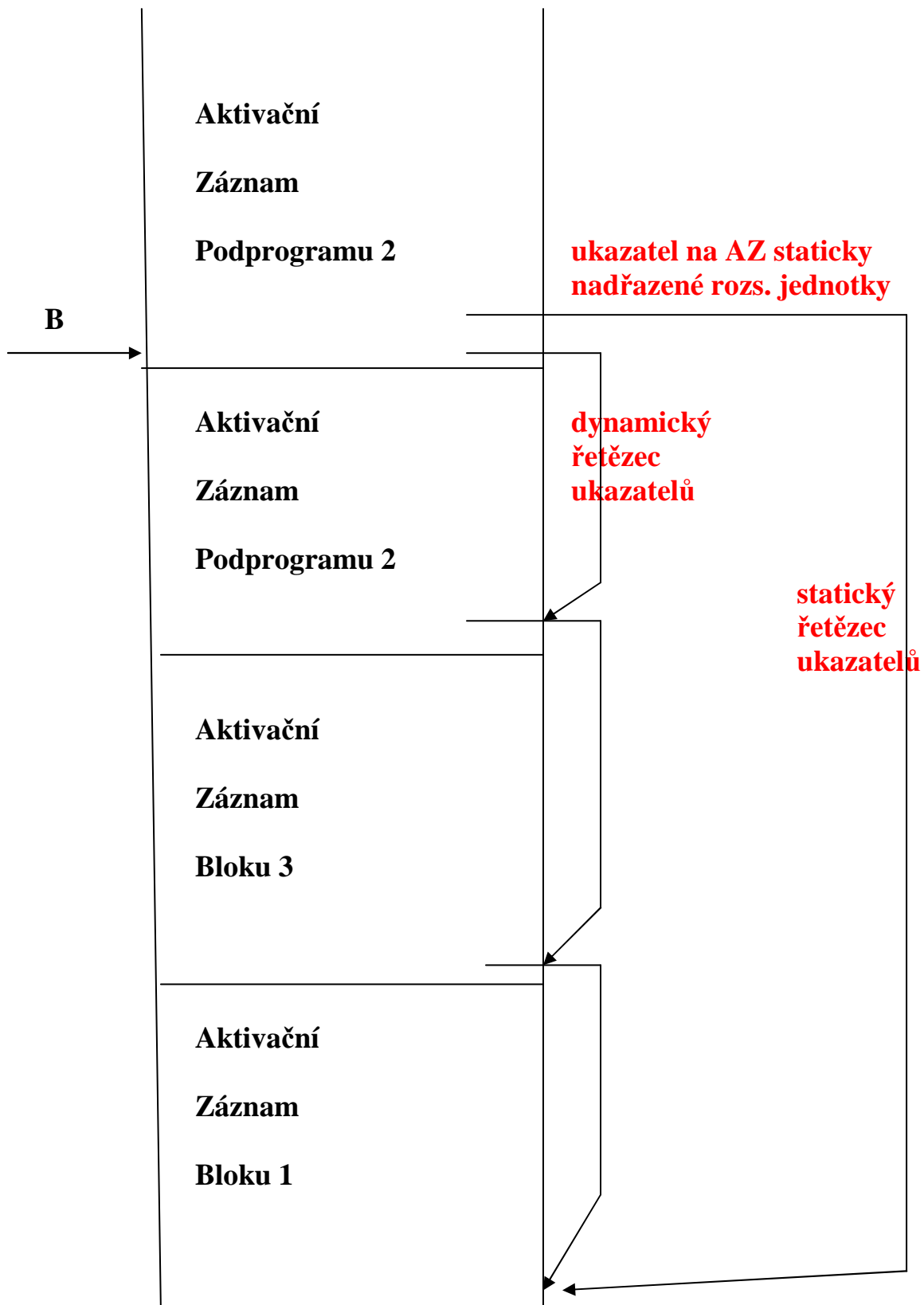
**Jaké bude uspořádání aktivačních záznamů při rekurz. vyvolání podpr.2 ?
 Pro rušení AZ při výstupu z jednotky potřebujeme tzv dynamický ukazatel**



Obr. Zásobník při rekurzivním volání podprogramu 2

B je registr ukazující na vrcholový AZ

Potřebujeme ještě vyřešit přístup k nelokálním proměnným při statickém = lexikálním rozsahu platnosti jmen. To řeší tzv. řetězec statických ukazatelů



Obr. Zásobník se statickým (ukazuje na lexikálně nadřazený AZ) a dynamickým řetězcem ukazatelů při rekurzivním volání podprogramu 2
 Pozn.: Statický uk. je nakreslen (pro přehlednost) jen u AZ rek. volání podpr.2

Vytváření řetězců ukazatelů

Nechť AZ má tvar:

pomocné proměnné
Lokální proměnné
Parametry
Funkční hodnota
Statický ukazatel
Dynamický ukazatel
Návratová adresa

↑
směr
růstu

Uvažujme zásobník Z, s vrcholem (nejvyšší zabranou adresou) T, n je hladina deklarace, m je hladina volání

Při vstupu do rozsahové jednotky (vyvolání podprogramu nebo vstupu výpočtu do bloku = Aktivace rozsahové jednotky):

- A1) $Z[T + 1] \leftarrow$ návratová adresa /* pouze u podprogramů*/
- A2) $Z[T + 2] \leftarrow B$ /*nastavení dynamického ukazatele*/
- A3) $Z[T + 3] \leftarrow B$
For i \leftarrow 1 to m - n do $Z[T + 3] \leftarrow Z[Z[T + 3] + 2]$ /*nastavení statického ukazatele*/
- A4) $B \leftarrow T + 1$ /*nastavení báze registru*/
- A5) $T \leftarrow T +$ velikost aktivačního záznamu
- A6) skok na první instrukci podprogramu a uložení do Z údajů o skutečných parametrech /*pouze u podprogramů*/

Pozn. Je-li podprogram překládán odděleně (neznámá velikost jeho AZ), pak je úprava T provedena až na začátku volaného podprogramu.

Při výstupu z rozsahové jednotky (Návrat z podprogramu nebo průchod koncem bloku):

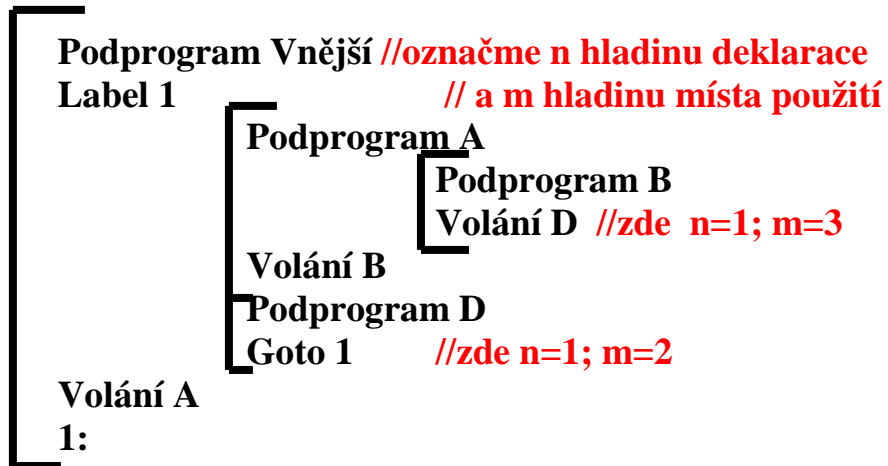
- N1) $T \leftarrow B - 1$
- N2) $B \leftarrow Z[B + 1]$
- N3) skok na adresu uloženou v $Z[T + 1]$ /*pouze u podprogramů*/

Výstup z rozsahové jednotky nelokálním skokem (hladina n deklarace návěští je menší než hladina m místa s příkazem skoku)

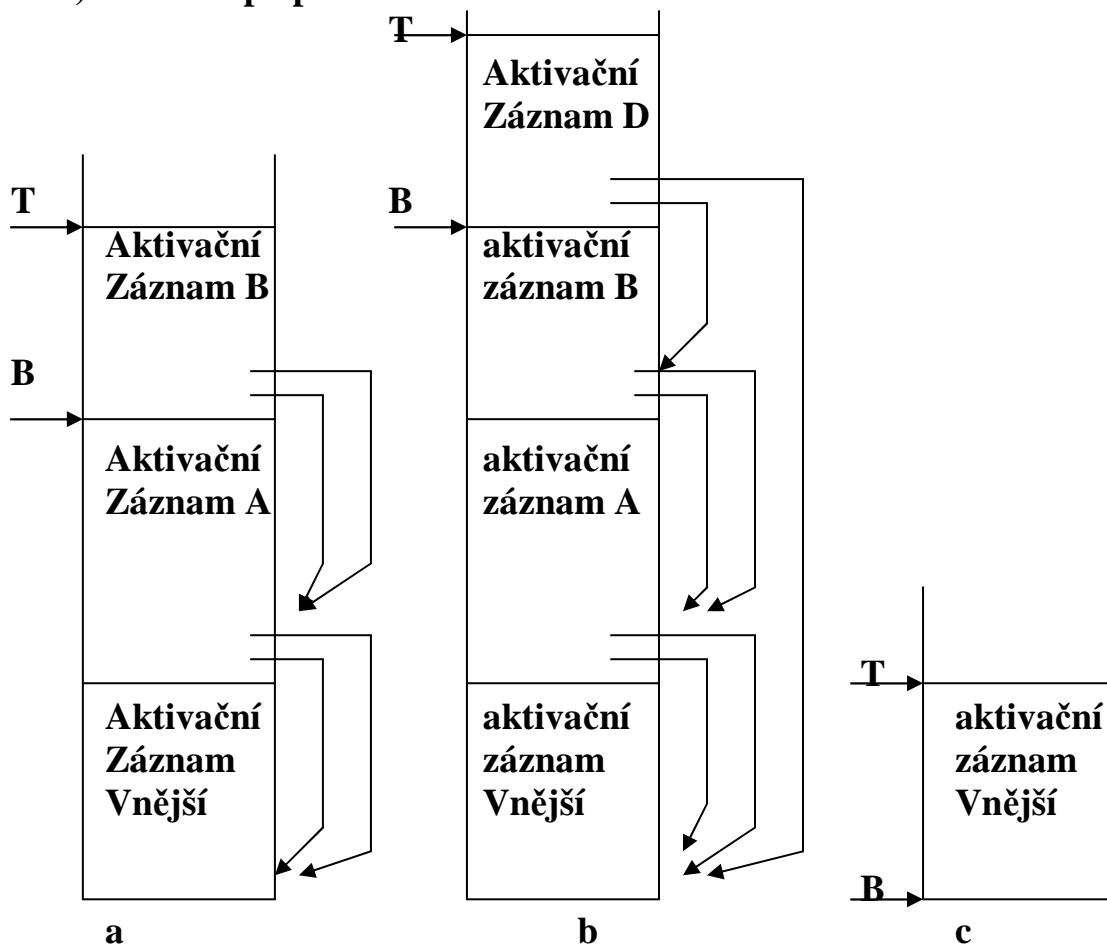
Vždy platí $n \leq m$

- S1) for i \leftarrow 1 to m - n do { Pom \leftarrow B
repeat $T \leftarrow B - 1$
 $B \leftarrow Z[B + 1]$
until $B \neq Z[POM + 2]$
}
- S2) skok na adresu, kterou návěští představuje

Př.2



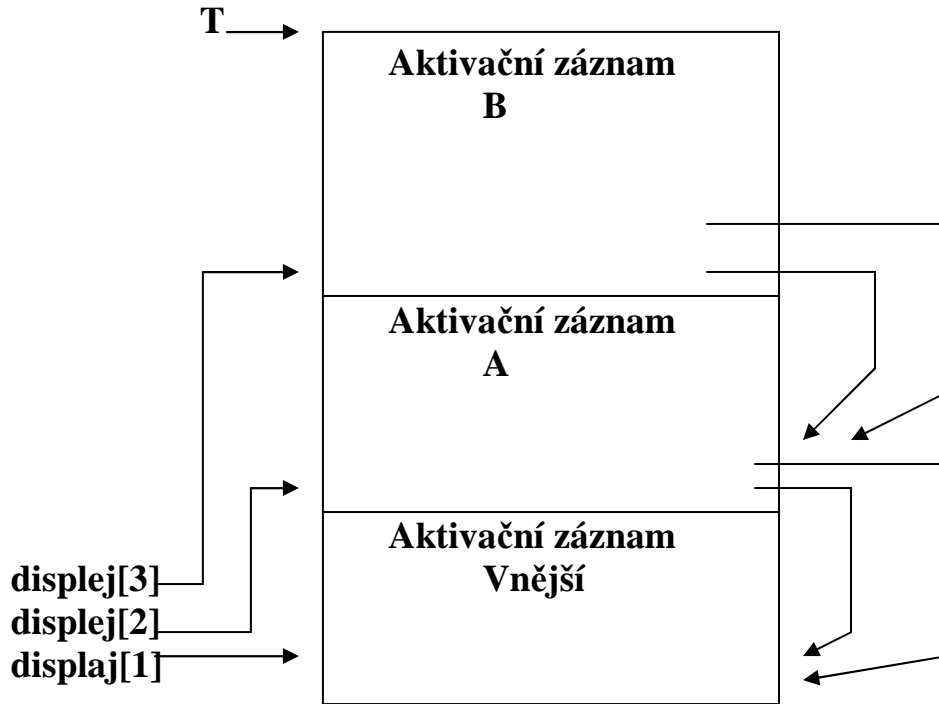
- a) obsah Z při provádění B, před voláním D,
- b) obsah Z po vyvolání D, před provedením nelokálního skoku,
- c) obsah Z po provedení skoku.



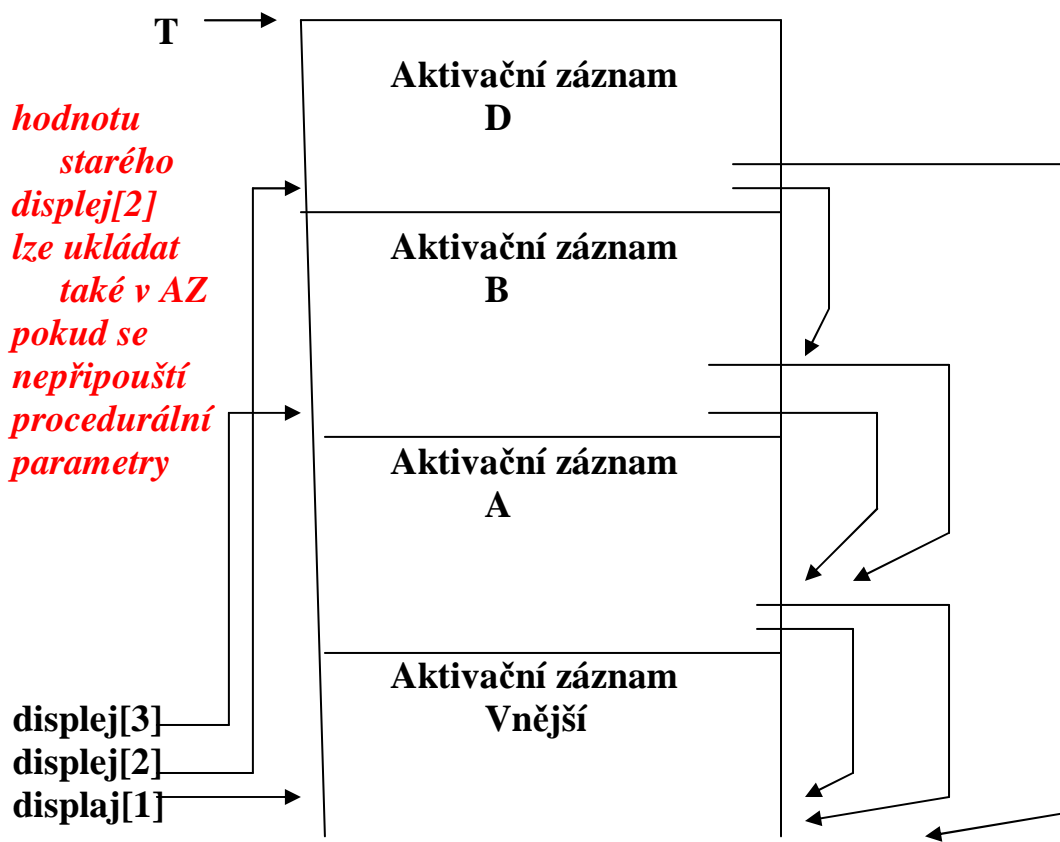
ad b) je to stav v okamžiku volání podpr. s hladinou deklarace 1, volaného v místě s hladinou 3

ad c) stav po výskoku z hladiny 2 do místa s hladinou 1

**Zrychlení přístupu k nelokálním proměnným
(pomocí vektoru ukazatelů displej[i], kde i je hladina rozs. jedn.)**



Obr. Stav Z při výpočtu B z př.2

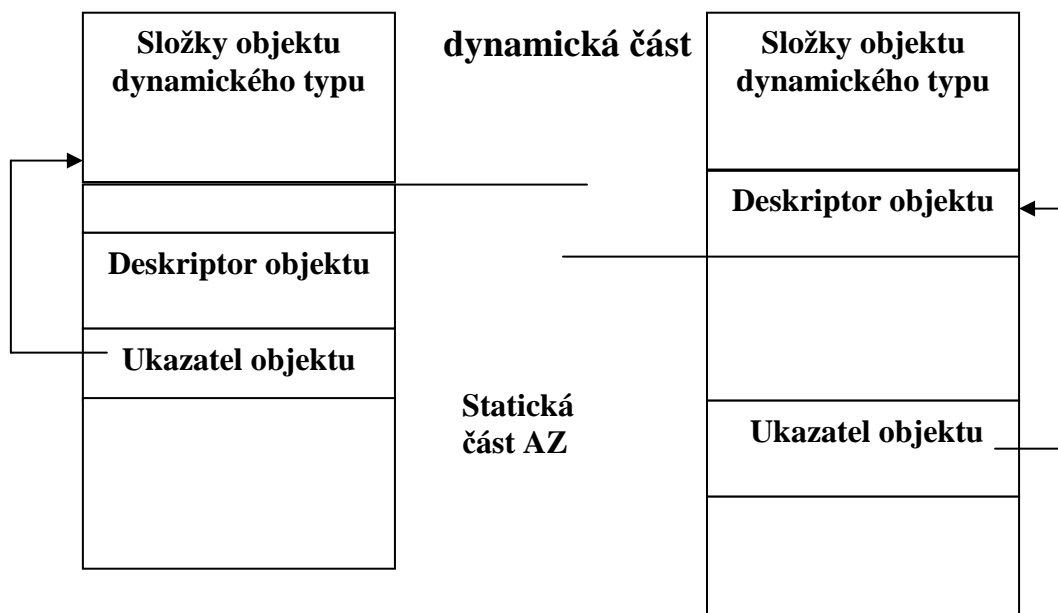


Obr. Stav Z při výpočtu D z př.2

Dynamická adresa proměnné je dvojice $(n, p) = \text{displej}[n] + p$

Objekty s dynamickými typy (typicky pole s proměnnými mezemi)

Možnosti struktury aktivačního záznamu s objektem dynamického typu



Deskriptor se vytvoří při překladu, uchovat se ale musí i při výpočtu

Př.3 Aktivačního záznamu s objekty dynamického typu

```
podprogram PRIKLAD;
  int i, j ;
  int A(m .. n);
  int B(p .. q, r .. s, );
```

dynamická část	místo pro prvky pole B místo pro prvky pole B
statická část	Descriptor B Ukazatel na prvky B Descriptor A Ukazatel na prvky A i j Parametry podprogramu Statický ukazatel Dynamický ukazatel Návrátová adresa

Předávání parametrů podprogramům

- hodnotou (C, C++, Java, C#) formální parametr je lokální proměnnou do níž se předá hodnota
- odkazem (C, C++ je-li parametrem pointer, objektové parametry Javy, C# označené ref) předá informaci o umístění skutečného parametru
- výsledkem - formální parametr je lokální proměnnou z níž se předá hodnota do skutečného parametru před návratem z podprogramu slouží jako výstupní parametr
- hodnotou výsledkem (novější Fortran) - kombinace
- jménem – má efekt textové substituce (jako historická zajímavost)
- v případě strukturovaných parametrů
 - jsou-li to statické typy ⇒ předá se adresa prvního prvku
 - jsou-li to dynamické typy ⇒ předá se ukazatel na descriptor
- je-li parametrem podprogram
 - u jazyků nedovolujících hnízdění podprogramů ⇒ předá se adresa začátku = pointer
 - u jazyků dovolujících hnízdění podprogramů ⇒
 - spolu s adresou musí předat i platné prostředí. Jsou různé možnosti co považovat za platné prostředí:
 - mělká vazba ⇒ platné je prostředí v němž se nachází volání formálního podprogramu**
 - hluboká vazba ⇒ platné je prostředí kde je předávaný podprogram definován**
 - ad hoc vazba ⇒ platné je prostředí kde je vydán příkaz volání podprogramu jež má za parametr podprogram**

Př.4

```
Podprogram P1() {  
  Prom x ;  
  Podprogram P2 () {  
    Vytiskni (x) ; /*co se tady tiskne?*/  
  };  
  Podprogram P3 () {  
    Prom x ;  
    x ← 3;  
    P4(P2) ;  
  };  
  Podprogram P4( podprogram Px ) {  
    Prom x ;  
    x ← 4;  
    call Px();  
  }  
  x←-1;  
  P3();  
}
```

prostředí, kde je předávaný podprogram definován

prostředí, kde je vydán příkaz volání s parametrem podprog.

prostředí, v němž je volán formální podprogram

Při mělké vazbě se tiskne ... ?

Při hluboké vazbě se tiskne ... ?

Při ad hoc vazbě se tiskne ... ?

Př.5

Předpokládejme hlubokou vazbu. Co se vytiskne po spuštění procedury Vnější?

```
podprogram Vnejsi; {  
  prom i:int;  
  podprogram P( podprogram FP; prom k:int;) {  
    prom i:int;  
    i←k+1; FP(); tisk(i);  
  }  
  podprogram Q(i:int);  
  podprogram R () {  
    Tisk(i);  
  }  
  P(R,i);  
}  
i← 0; Q(i+1);  
}
```

Stav před vyvoláním a po vyvolání formálního poprogramu FP z př.5

15			
T → 14	hodnota i=2	lokální proměnná	
13	adresa k=7		
12	statické prostředí R=4	formální parametry	
11	adresa začátku R		aktivační záznam P
10	statický ukazatel =0		
9	dynamický ukazatel =4		
B → 8	návratová adresa P		
7	hodnota i=1	formální parametr	
6	statický ukazatel =0		aktivační záznam Q
5	dynamický ukazatel =0		
4	návratová adresa Q		
3	i=0	lokální parametr	
2			aktivační záznam
1			
0	návratová adresa Vnější		Vnější

T → 18			
17	stat. ukazatel R=4		<i>aktivační záznam R</i>
16	dynam. ukaz. R=0		
B → 15	návratová adr. R		
T → 14	hodnota i=2	lokální proměnná	
13	adresa k=7		
12	statické prostředí R=4	formální parametry	
11	adresa začátku R		aktivační záznam P
10	statický ukazatel =0		
9	dynamický ukazatel =4		
B → 8	návratová adresa P		
7	hodnota i=1	formální parametr	
6	statický ukazatel =0		aktivační záznam Q
5	dynamický ukazatel =0		
4	návratová adresa Q		
3	i=0	lokální parametr	
2			aktivační záznam
1			
0	návratová adresa Vnější		Vnější

Přidělování paměti pro paralelní výpočty

Pro uložení AZ paralelního výpočtu nutno použít haldu nebo zobecněný zásobník

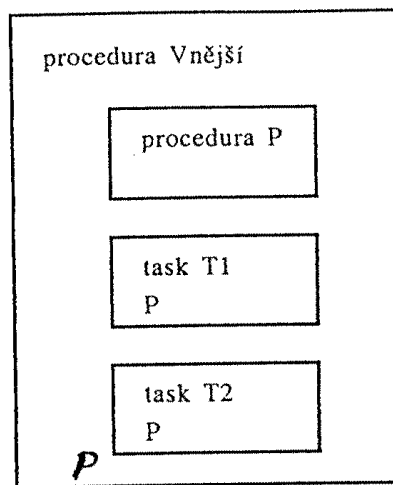
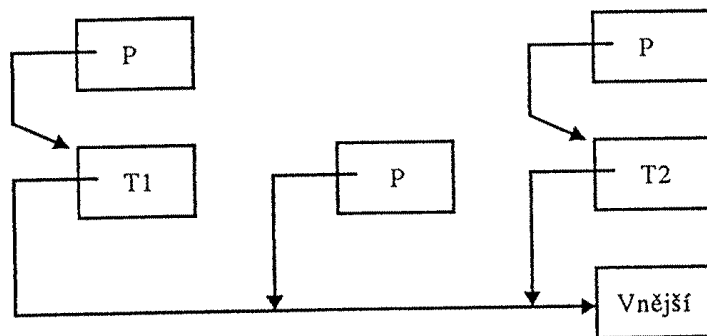
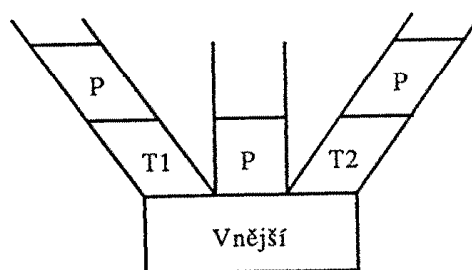


Schéma vnoření bloků programu



Struktura aktivačních záznamů při paralelním výpočtu



Uložení aktivačních záznamů ve zobecněném zásobníku

Př v javovském prostředí

