Principy překladačů

Běhová podpora

Jakub Yaghob

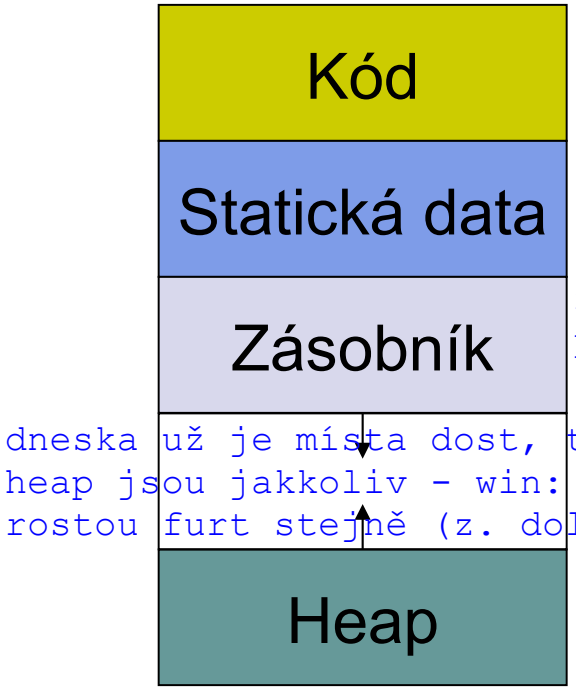
Běhová podpora

- Statická podpora jazyka
  - Překladač
  - Interface na knihovny
    - Hlavičkové soubory
- Dynamická podpora jazyka
  - Prostředí běhu programu
    - Rozdělení paměti
    - Stav paměti před spuštěním
    - Konstruktory, destruktory globálních objektů
  - Knihovny `interface` `C: stačí preprocesor kterej načte *.h`
  - Volací konvence `často podobně jako OS; rozhraní mezi usercode a OS`

Rozdělení paměti

registr ukazující na vrchol zásobníku, překladač předem spočítá, o kolik se to má zvětšit/zmenšit -> za běhu 1 instrukce (oproti heapu, tam musim hledat místo)

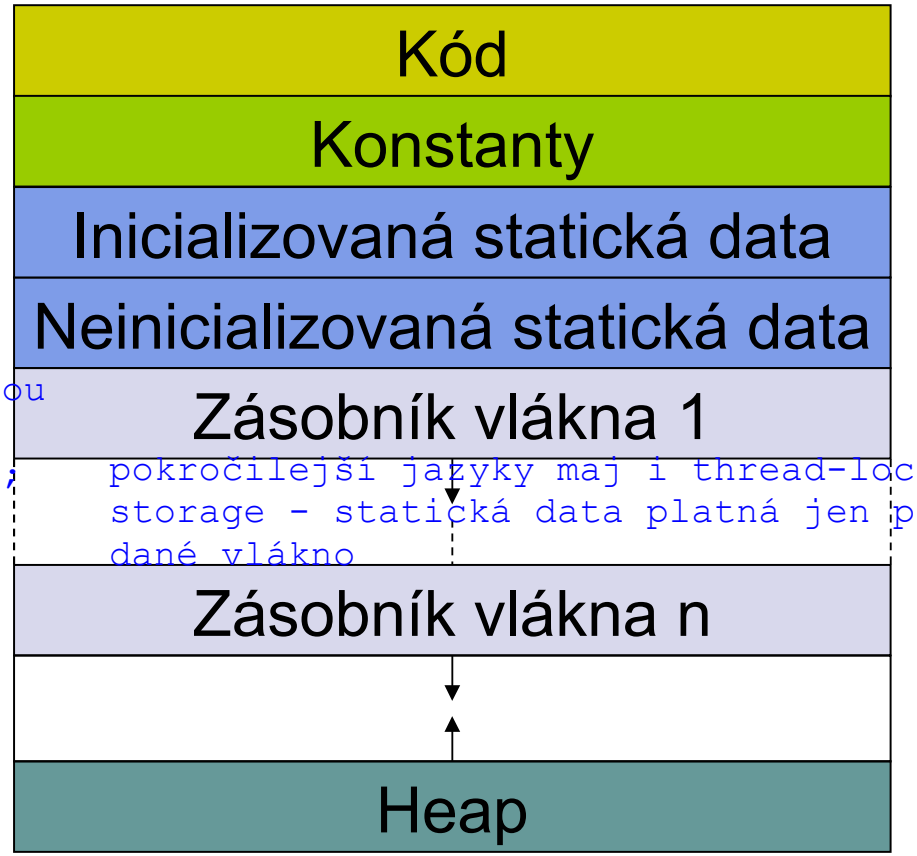
- Rozdělení paměti za běhu v procedurálních programech



globální prom.
řetězcové konst
apod.

nová data vzniknou při volání fce, zaniknou při ret;

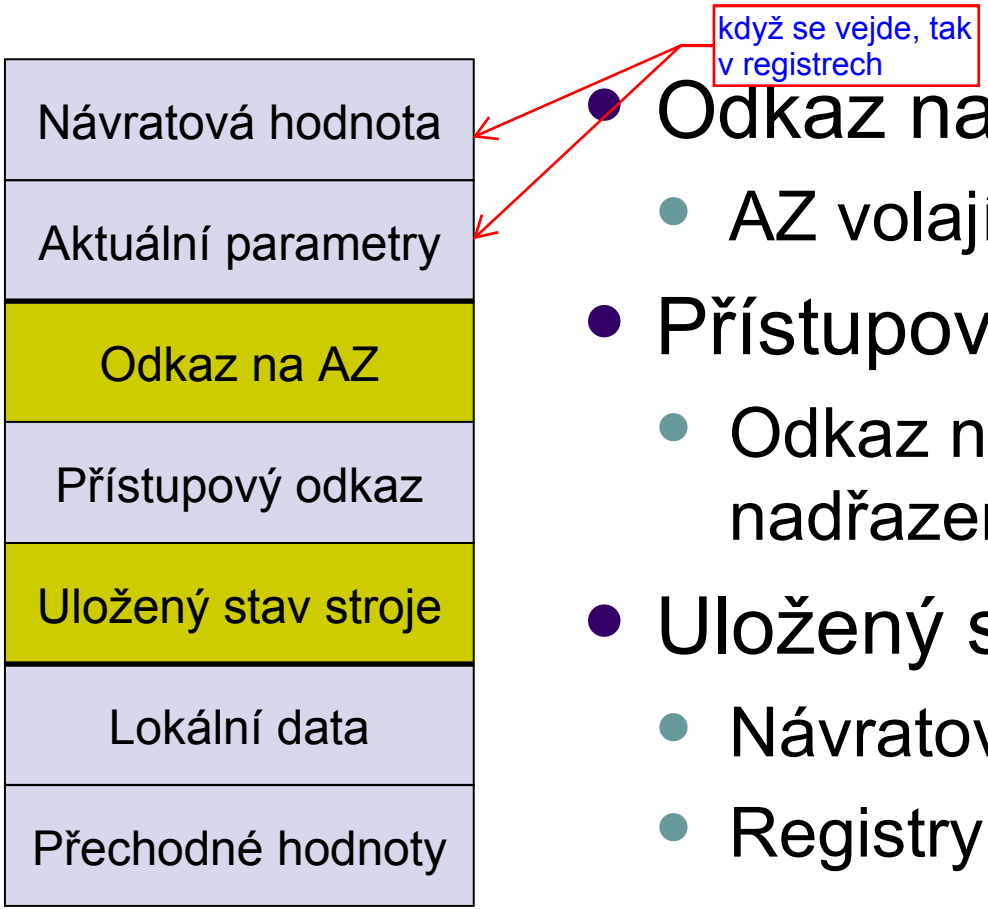
dneska už je místa dost, tak zásobník a heap jsou jakkoliv - win: jsou obráceně, rostou furt stejně (z. dolu, h. nahoru)



pokročilejší jazyky maj i thread-local storage - statická data platná jen pro dané vlákno

Aktivační záznam

= úsek na zásobníku,
něco vyplní volající, něco volaný



- Odkaz na AZ = předchozí Aktivační záznam, týká se volající procedury
  - AZ volající funkce
- Přístupový odkaz
  - Odkaz na data sémanticky nadřazených funkcí

jazyky co maj vnořený procedury (např. PL/SQL =staticky nadřazená

- Uložený stav stroje
  - Návratová adresa do kódu
  - Registry

Volací konvence

= znetvoření; linkry jsou předpotopní, formát taky
long f1(int i, ...) -> ?f1@@YAJH...@@@Z
aby se předaly další info (v C ok)

- Volací konvence
  - Mandlování veřejných jmen
  - Mechanismus volání funkcí a procedur
    - Zodpovědnost za úklid
  - Způsob předávání parametrů
    - Registry, zásobník
    - Pořadí ukládání parametrů na zásobník
  - Způsob předávání návratové hodnoty funkcí
    - Registry, zásobník
  - Požadavky na uchování stavu registrů
    - Registry pro registrové proměnné

kdo zodpovídá za úklid parametrů, apod. - obvykle podle jazyka - C: voljaící, Pascal: volaná

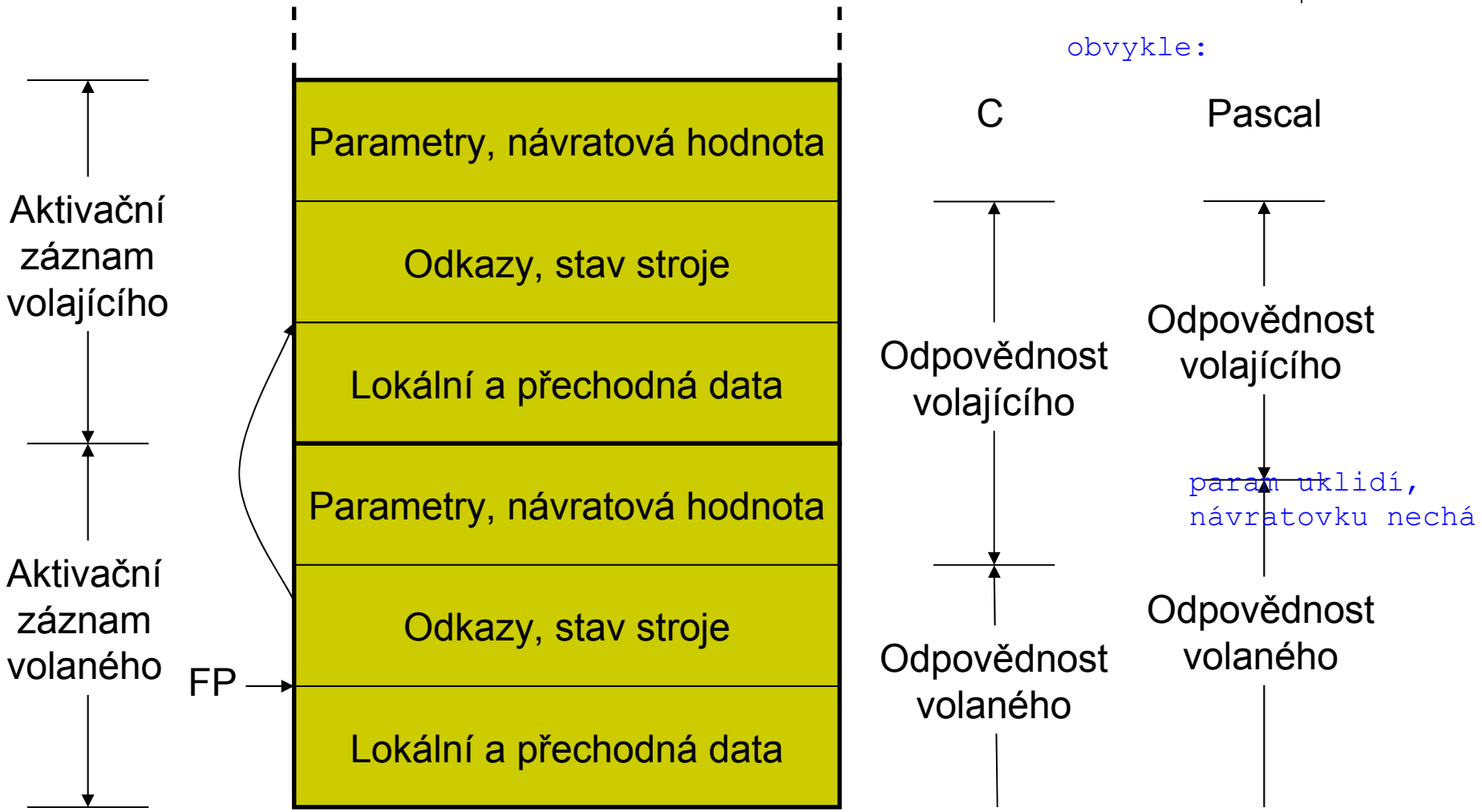
třeba první 4 parametry co se vlezou do registru se předávaj registrem

zleva doprava/zprava doleva
C++: zprava doleva =>
proměnné počty param
volaná musí nějak vědět,
co dostává za proměnné para

některý registry se musej zachovat
některý se musej přepsat původním kontextem

ABI: binary interface - pro knihovny:
výhoda: přenoitelnost (mezi překladačema),
nevýhoda: pro některé jazyky by možná bylo vhodnější to udělat jinak

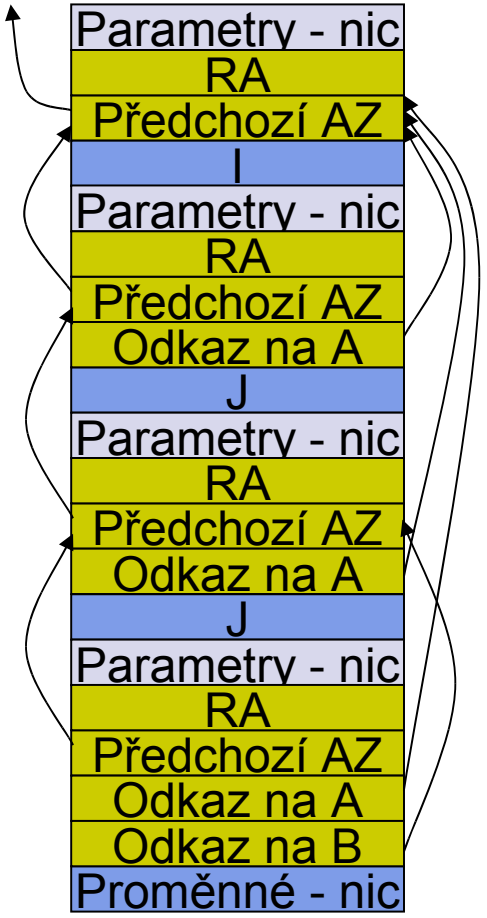
Mechanismus volání funkcí



Přístup k sémanticky nadřazeným proměnným

- Pascal
  - Vnořené funkce

```
procedure A;
var I:integer;
   procedure B;
   var J:integer;
        procedure C;
        begin …I+J… end;
   begin
        if I>0 B else C;
   end;
begin B end;
```



Předávání parametrů

- Předávání hodnotou
  - Aktuální hodnota parametru vypočtena a okopírována
  - Vstupní parametry, chová se jako lokální proměnná
  - C, „nevařené“ parametry v Pascalu
- Předávání referencí
  - Volající předá adresu na místo v paměti
  - Vstupně/výstupní parametry
  - & v C++, „vařené“ parametry v Pascalu
- Předávání jménem
  - Chová se jako makro – aktuální výraz se „inlinuje“ na místech použití `SQL`

Dynamická paměť

- Alokační algoritmy
  - Spojité bloky volné velikosti
- Garbage collector
  - Implicitní dealokace
  - Čítače odkazů
  - Značkování
    - V jistém okamžiku se zastaví běh programu
    - Všechny ukazatele musí být známy včetně typů
    - Všechny bloky se označí jako nepoužité
    - Prochází se rekurzivně ukazatele a značkují se použité bloky
    - Nepoužité se odstraní
    - Lze i setřepat