Sistemske interakcije virtuelne memorije

Literatura: Hennessy & Patterson, B.1–B.4

Četiri pitanja memorijske hijerarhije

- Gde se memorijski blok može smestiti na višem nivou?
- Kako pronaći blok na višem nivou?
- Koji blok zameniti prilikom promašaja?
- Šta se dešava prilikom pisanja vrednosti (ili, kakva je politika pisanja)?
- Ova pitanja se podjednako mogu postaviti i za keš i za virtuelnu memoriju; prvo ćemo diskutovati keš

Keš: pitanja hijerarhije

- Kod keša, blok = linija (64 bajta)
- Prvo pitanje (gde se linija može smestiti) je već diskutovano: asocijativnost
- Drugo pitanje (kako naći liniju) takođe je postavljeno, u vezi s prvim, ali ćemo ga dodatno obraditi zbog interakcije s virtuelnom memorijom
- Za preostala dva pitanja postoje uočljive paralele između keširanja i virtuelne memorije

Keš: koju liniju zameniti?

- Ako ima slobodnih, nije problem
- U suprotnom, nekoliko strategija:
  - (pseudo)slučajni izbor sa ravnomernom raspodelom
  - LRU (Least Recently Used) najmanje skoro korišćena; oslanja se na logičku posledicu lokaliteta —ako linija nije skoro korišćena, verovatno neće biti skoro potrebna
  - FIFO (First In, First Out) ali u **obrnutom** redosledu: uzima se najstarija linija kao aproksimacija najmanje skoro korišćene. FIFO se lakše implementira nego LRU

Keš: šta se dešava prilikom pisanja?

- Kod realnih programa čitanje memorije je češće od pisanja, otprilike 5:2 u nekim merenjima
- Dodatno, svako prihvatanje instrukcije je čitanje
- Čitav memorijski podsistem se može praktično optimizovati za čitanje
- Pisanje se mora obavljati pažljivije od čitanja:
  - vrednosti se ne smeju zapisivati dok niste sigurni da je prâva linija u kešu
  - mora se strogo voditi računa o veličini promena

Keš: politika pisanja

- Prolazno (write through): sve vrednosti se odmah zapisuju i u keš i na nižem nivou
- Zadržano (write back): vrednosti se isprva zapisuju samo u keš, mogu se zapisati niže prilikom zamene linije ili periodično
- Svaka od politika ima svoje prednosti i mane
- Po potrebi, mogu se kombinovati

Keš: karakteristike politika pisanja

- **Prolazno** pisanje se jednostavnije implementira
- Takođe, sadržaj keša je uvek čist i ažuran
- Ovo naročito ima prednosti kod višeprocesorskih sistema
- Kod **zadržanog** pisanja:
  - pisanje se obavlja brzinom keša, a ne nižeg nivoa memorije
  - potencijalno više zapisa u keš biva pokriveno jednim zapisom u niži nivo
  - na taj način, ova politika zahteva manji propusni opseg memorijskog podsistema

Keš i virtuelna memorija: interakcija

- Pošto postoje virtuelne i fizičke adrese, pitanje je kako ih razrešavati prilikom traženja linije u kešu
- Podsećanje, kad se odbaci ofset (u okviru linije) deo ostatka adrese služi za izbor skupa (kod skupovno asocijativnog keša)—označili smo sa S
- Taj izbor se zove indeksiranje, koje onda može biti virtuelno ili fizičko
- Ostatak adrese je tag, takođe može biti virtuelni ili fizički

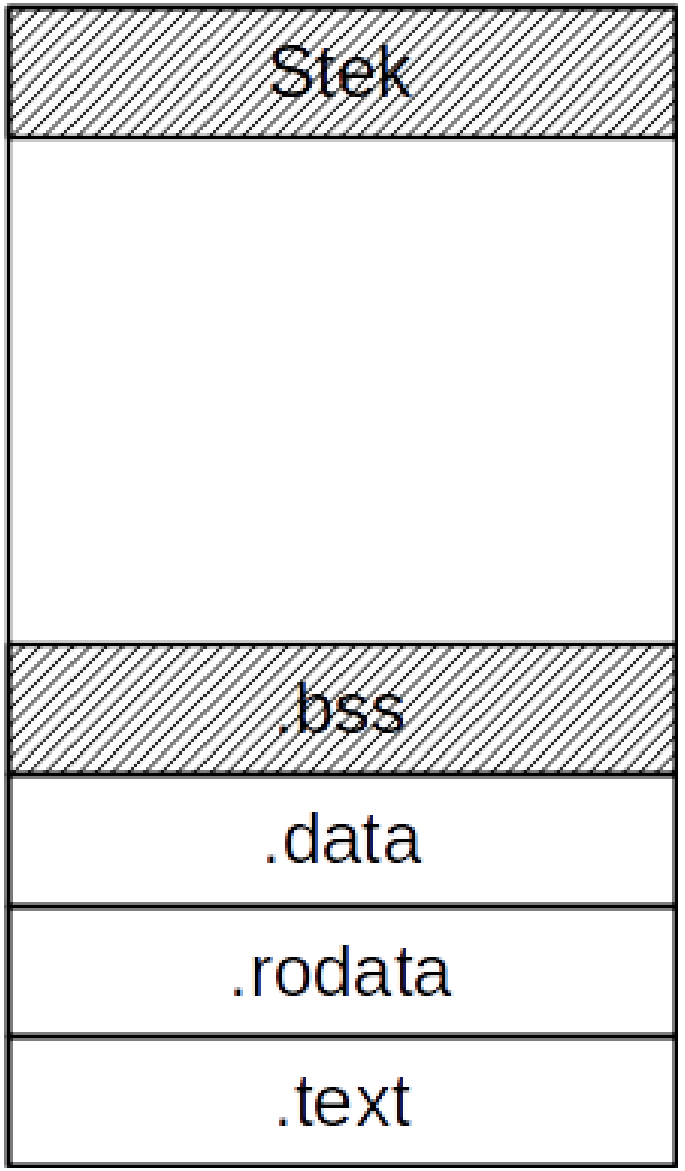
Kombinacije indeksiranja i tagovanja

- Četiri moguće: PIPT, PIVT, VIPT, VIVT
- Problem fizičkog indeksiranja je što se virtuelna adresa mora prvo prevesti, što unosi kašnjenje, ali je ukupna implementacija jednostavnija
- Kod VIPT, indeksiranje se može pokrenuti u paraleli s prevođenjem, koje je potrebno za tag
- Virtuelno tagovanje se u praksi ne koristi zbog veće verovatnoće postojanja duplikata
- Jedna uobičajena implementacija: L1 je VIPT, L2/L3 su PIPT

Četiri pitanja primenjena na virtuelnu memoriju

- Ovde je blok = stranica
- Gde se stranica smešta u radnoj memoriji? Obično gde god ima slobodnog mesta (ekvivalent potpune asocijativnosti) zbog velikog troška čitanja sadržaja sekundarne memorije
- Kako naći stranicu? → Tabela stranica
- Koju stranicu zameniti? Najčešće LRU, ali postoji problem masovnog sekvencijalnog čitanja
- Politika pisanja? Isključivo zadržano

Idealizovana memorijska mapa procesa



Segmenti memorijske mape

- Svaki od segmenata ima određene privilegije:
  - R: samo čitanje
  - W: čitanje i pisanje
  - X: izvršavanje
- **.text**: instrukcije, R,X
- **.rodata**: konstante, R
- **.data**: promenljivi, inicijalizovani podaci, W
- **.bss**: promeljivi podaci inicijalizovani na vrednost 0, W (Block Started by Symbol), vode se posebno jer ne zauzimaju mesto u izvršnom fajlu

Primer memorijske mape

- U izvršnom fajlu: `objdump -h /bin/bash`

```
13 .text    41bb20
15 .rodata  4a6da0
24 .data    6de6a0
25 .bss     6e6aa0
```

- U virtuelnoj memoriji: `/proc/$$/maps`

```
400000-4dd000 r-xp
6dd000-6de000 r--p
6de000-6e7000 rw-p
```

Realnija situacija

- Raspored u virtuelnoj memorijskoj mapi zavisi od prevodioca
- Specifični raspored adresa neće sam po sebi uticati na performanse (tj. nije naročito bitno da li program počinje od 0x400000 ili 0x500000)
- Postoje tehnike koje namerno menjaju položaj segmenata u memoriji, kao što je ASLR (Address Space Layout Randomization)
- Relokacija i indirekcija svejedno moraju postojati zbog dinamičkog povezivanja

Statički i dinamički povezani programi

- Statički program sadrži u sebi sve komponente potrebne za izvršavanje
- Prednost: samodovoljan, pogodniji za distribuciju i instalaciju
- Mana: veliki broj programa sadrži delove koda koji su zajednički, iz sistemskih biblioteka
  - ovo povećava prostor za skladištenje (disk/flash), danas nije toliko bitno
  - povećan je utrošak memorije prilikom istovremenog izvršavanja
  - komplikuje ispravku grešaka u zajedničkom kodu

Dinamičke biblioteke

- Biblioteke (skupovi zajedničkih funkcija) prevedeni i povezani na poseban način
- Koriste poziciono nezavistan kod, tako da mogu biti virtuelno mapirane na različite adrese u okviru različitih procesa, a da koriste istu fizičku memoriju
- Zbog toga moraju imati posebnu podršku za relokaciju i indirekciju poziva
- Konačno povezivanje i pokretanje programa radi **dinamički linker**

Primer dinamički povezanog progama

- Komanda: `ldd /bin/bash`

```
linux-vdso.so.1 =>  (0x00007ffe5e176000)
libtinfo.so.5 => /lib64/libtinfo.so.5 (0x00007ff5146e9000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007ff5144e5000)
libc.so.6 => /lib64/libc.so.6 (0x00007ff514118000)
/lib64/ld-linux-x86-64.so.2 (0x00007ff514913000)
```

- Prvi red nema mapiranje jer je u pitanju specijalna dinamička biblioteka
- Poslednji red je dinamički linker

Primer deljenja fizičke memorije

- Dva procesa koja koriste istu dinamičku biblioteku:

```
7f90ed0f7000 … r-xp … libc-2.17.so
7f4a184f4000 … r-xp … libc-2.17.so
```

- Prevod prve virtuelne adrese u fizičku:

```
Vaddr: 0x7f90ed0f7000, Page_size: 4096
PFN: 0x36a97
```

- Prevod druge virtuelne adrese u fizičku:

```
Vaddr: 0x7f4a184f4000, Page_size: 4096
PFN: 0x36a97
```

- PFN je Page Frame Number, fizička stranica