Višeprocesni rad i virtuelizacija

Terminologija

- Proces: kontekst za izvršavanje programskog koda
  - obično se vezuje za pojedinačni korisnički program, ali može se odnositi i na druge vrste organizacije
  - uključuje sve što je potrebno kao okruženje za izvršavanje
    - memorijska mapa
    - pristup sistemskim resursima (niži nivo)
    - prisutp sistemskim servisima (viši nivo)
- Proces se lako mapira na sekvencijalni model izvršavanja

Terminologija (2)

- Šta ako nam treba više procesa?
- **Konkurentno** izvršavanje
  - i dalje sekvencijalno, u jednom logičkom lancu instrukcija
  - vremenski podeljeno, tako da se svaki proces izvršava kratko vreme, pa pređe na drugi
  - dve varijante vremenskog deljenja:
    - kooperativno (*cooperative multitasking*)
    - diktirano (*preemptive multitasking*)
- I dalje se može relativno lako rezonovati o efektima izvršavanja
  - ali zgodno je imati izvesne garancije ponašanja radi uspešnijih apstrakcija i izbegavanja anomalija

Terminologija (3)

- **Paralelno** izvršavanje
  - više nezavisnih lanaca izvršavanja
  - svaki od tih lanaca može biti podržan:
    - potpuno nezavisnim fizičkim procesororom
    - integrisanim ali nezavisnim procesorskim jezgrom
    - skupom funkcionalnih jedinica u okviru jednog jezgra (Intel Hyper-threading)
- Uvodi potpuno nove probleme prilikom rezonovanja o efektima izvršavanja

Sistemska podrška za višeprocesni rad

- Svi procesi bi mogli da se izvršavaju u istom globalnom adresnom prostoru
  - na sistemima kao što je MS/PC-DOS i originalni MacOS to i jeste bio slučaj
- Problem: šta ako proces zbog greške ili iz zle namere poremeti memoriju drugog procesa?
  - već smo razmatrali: upravljanje memorijom/MMU
- Sledeći problem: šta ako proces iz istih razloga poremeti MMU tabele?
  - jedan nivo indirekcije do memorije i resursa drugih procesa

Privilegovani režim rada

- Uvodi se poseban, privilegovan, režim rada procesora
  - ovo deli način izvršavanja na najmanje dva dela: neprivilegovan i privilegovan
- U privilegovanom režimu moguće je pristupiti sistemskim resursima, koji postaju nedostupni u neprivilegovanom režimu
  - MMU tabele
  - prekidne tabele
  - U/I portovi
- Privilegovani kod se najčešće grupiše u celinu koja se zove **kernel**

Nivoi privilegija

- Privilegovani režim može imati više nivoa privilegija, u najprostijoj varijanti samo jedan
- Nivoi se u standardnoj nomenklaturi zovu
  - hijerarhijske oblasti zaštite (hierarchical protection domains)
  - prstenovi zaštite (protection rings)
- Recimo, sa četiri prstena (x86, amd64):
  - prsten 0: kernel
  - prsten 1: drajveri
  - prsten 2: privilegovani korisnički procesi
  - prsten 3: neprivilegovani korisnički procesi
- Linux, Windows NT+, moderni macOS: 0 + 3

Tipična memorijska mapa

Prelazak između nivoa

- Neprivilegovanom kodu su potrebne usluge sistema
- Mora postojati način da se (kontrolisano) pređe na izvršavanje privilegovanog koda
- Za x86/amd64:
  - softverski prekid: int 0x80
  - SYSENTER/SYSEXIT (x86, 32-bitni režim)
  - SYSCALL/SYSRET (amd64, 64-bitni režim)
- Menja memorijske mape!

Prelazak između procesa

- Ako je svaki korisnički proces slično mapiran, MMU tabele moraju da se menjaju prilikom prelaska na sledeći proces
- Isto važi i prilikom prelaska između neprivilegovanog i privilegovanog režima rada
  - korisnički proces ne sme da ni da vidi, a kamoli da menja kernelske strukture podataka
- Problem: TLB zapisi, jer je TLB globalni resurs

Rukovanje TLB zapisima

- Najjednostavnije: zaboraviti sve TLB zapise prilikom prelaska
  - ali neefikasno: primera radi, kernelske mape su uglavnom fiksne, a sistemski poziv ne mora da poremeti veliki broj mapiranja
- Malo bolje: omogućiti individualno zaboravljanje TLB zapisa
- Još bolje: proširiti tagove za TLB identifikatorima adresnih prostora
  - kernelski i korisnički prostori dobijaju dodatni (kratak) identifikator
  - prostor za identifikatore je jako mali, jer je TLB keš vrlo ograničen i kritičan resurs
  - čak i jedan dodatni bit popravlja situaciju

Dodatni motiv: virtuelizacija

- Apstrakcija čitave hardverske platforme, tako da jedna fizička mašina može da izvršava više virtuelnih mašina
- Ako je postupak efikasan, ovo je dobar način da se iskoriste hardverski resursi
- Svaka virtuelna mašina ima sopstveni operativni sistem
- Mora postojati komponenta koja se izvršava na osnovnom nivou i vodi računa o ostalim VM: zove se hipervizor (*hypervisor*) ili VMM (*Virtual Machine Monitor*)

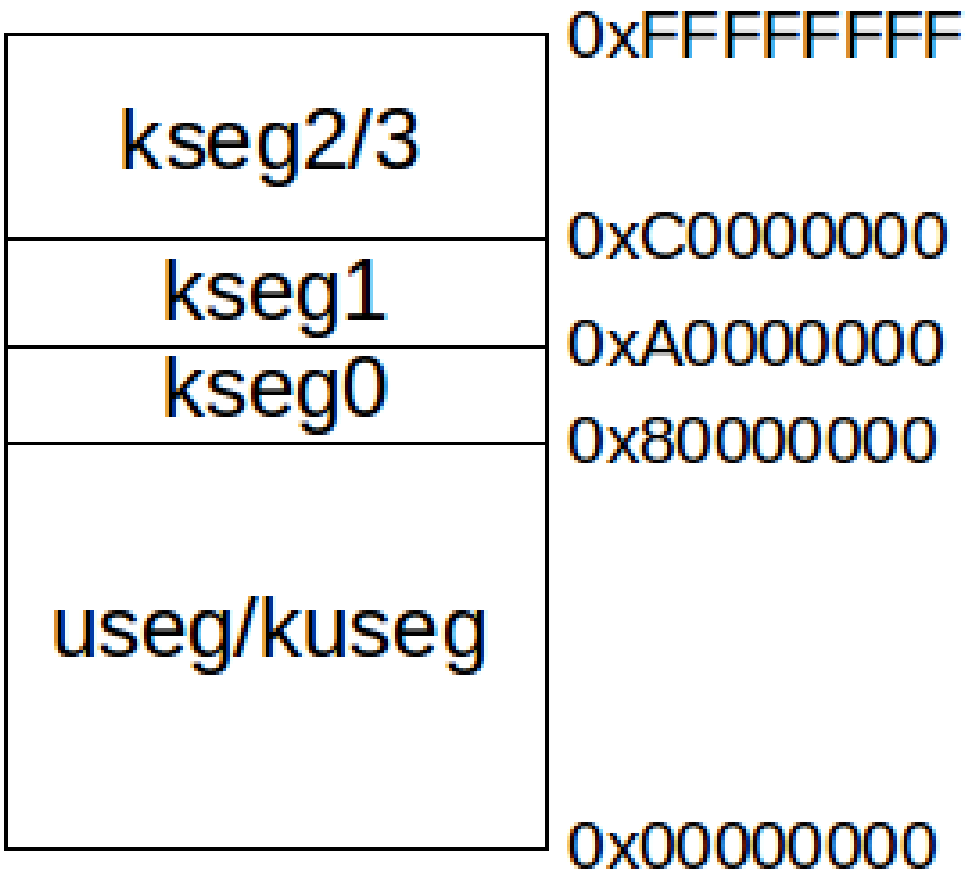
Zahtevi za virtuelizaciju (Popek/Goldberg)

- Ekvivalentnost/Vernost
  - program koji se izvršava pod hipervizorom ponaša se identično kao i na fizičkoj mašini
- Kontrola resursa/Bezbednost
  - hipervizor mora u potpunosti da kontroliše virtualizovane resurse
- Efikasnost/Performanse
  - statistički dominantan deo mašinskih instrukcija mora se izvršavati bez intevencije hipervizora

Načini virtuelizacije

- Puna virtuelizacija
  - potpuna imitacija hardverskog okruženja, operativni sistem unutar VM može se izvršavati bez modifikacija
- Hardverski potpomognuta virtuelizacija
  - omogućava punu virtuelizaciju uz pomoć hardverskih sklopova na fizičkom procesoru
- Paravirtuelizacija
  - zahteva modifikaciju operativnog sistema tako da eksplicitno sarađuje sa hipervizorom

Ilustracija: MIPS-32



MIPS-32, inicijalizacija

- Ima dva obavezna režima rada, kernelski i korisnički, sistem se posle resetovanja budi u kernelskom režimu
- U memorijskoj mapi pojedini segmenti se mogu referencirati samo iz privilegovanog režima (iznad 0x80000000)
- Takođe, segmenti se razlikuju prema tome da li se keširaju i mapiraju kroz MMU
  - kseg0: kernelski, nemapirani
  - kseg1: kernelski, nemapirani, nekeširan
  - kseg2/3: kernelski, mapirani
  - useg/kuseg: korisnički/kernelski, mapirani
- Kod ovog procesora, i keševi i MMU se moraju inicijalizovati softverski