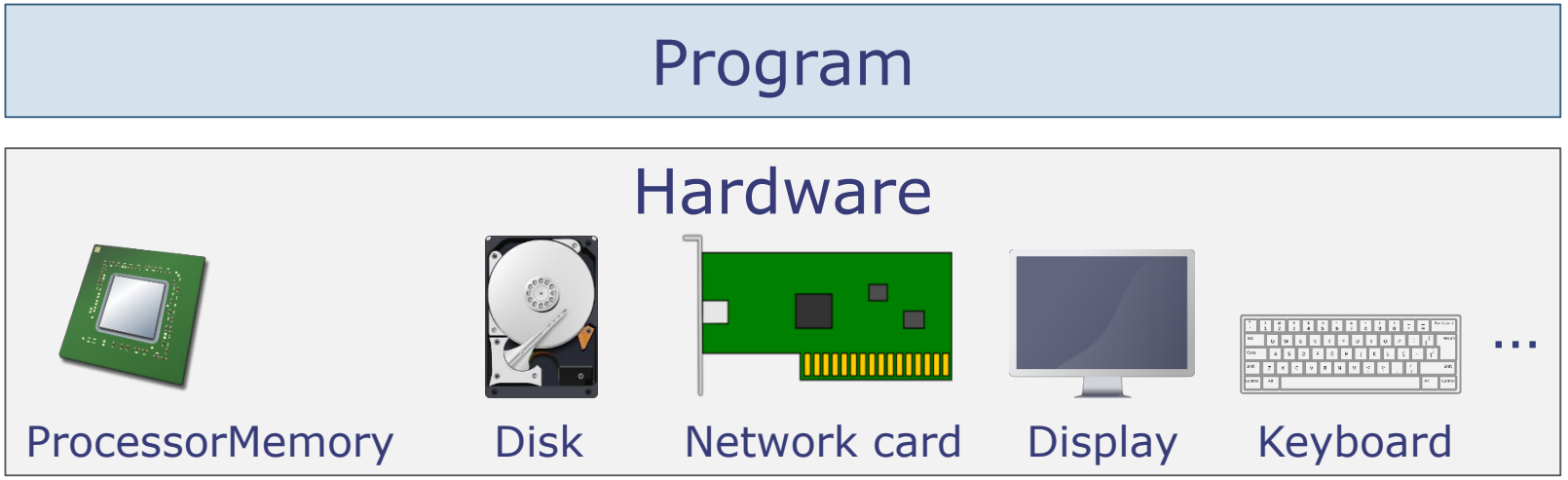
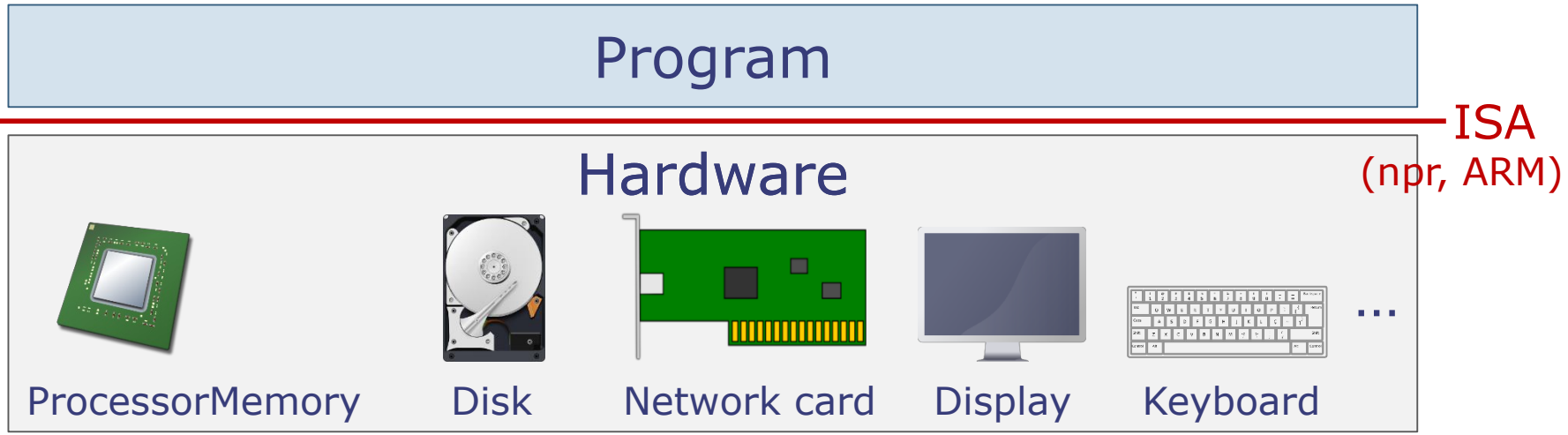
Operativni sistemi: Virtualne masine i izuzeci

Bez OS: Jedan program



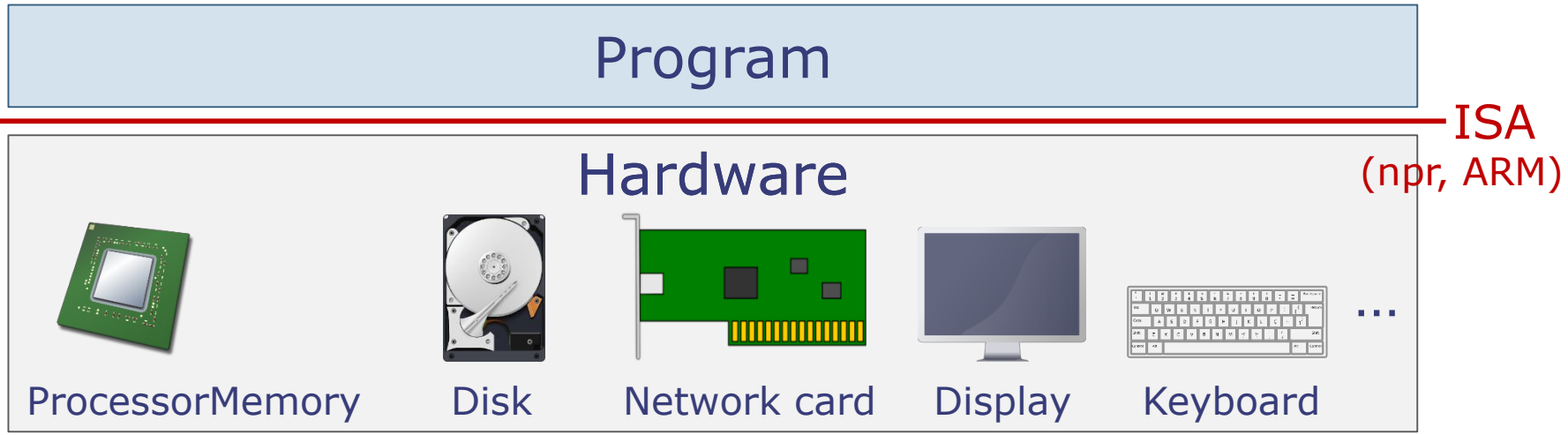
- HW izvrsava jedan program
- Ovaj program ima direktan i kompletan pristup svim svim hardverskim resursima na masini

Bez OS: Jedan program



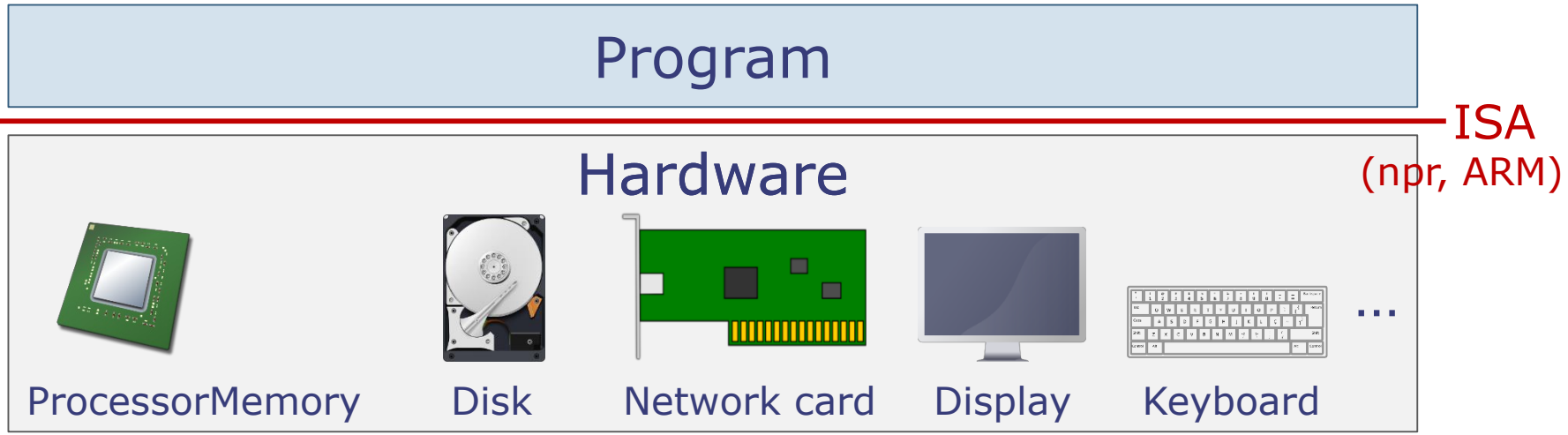
- HW izvrsava jedan program
- Ovaj program ima direktan i kompletan pristup svim svim hardverskim resursima na masini

Bez OS: Jedan program



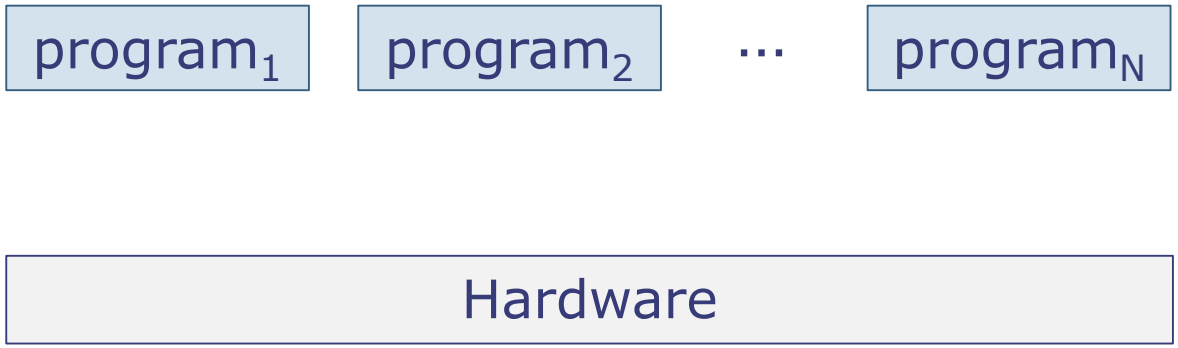
- HW izvrsava jedan program
- Ovaj program ima direktan i kompletan pristup svim svim hardverskim resursima na masini
- Arhitektura (*instruction set architecture* - ISA) je interfejs izmedju HW i SW

Bez OS: Jedan program



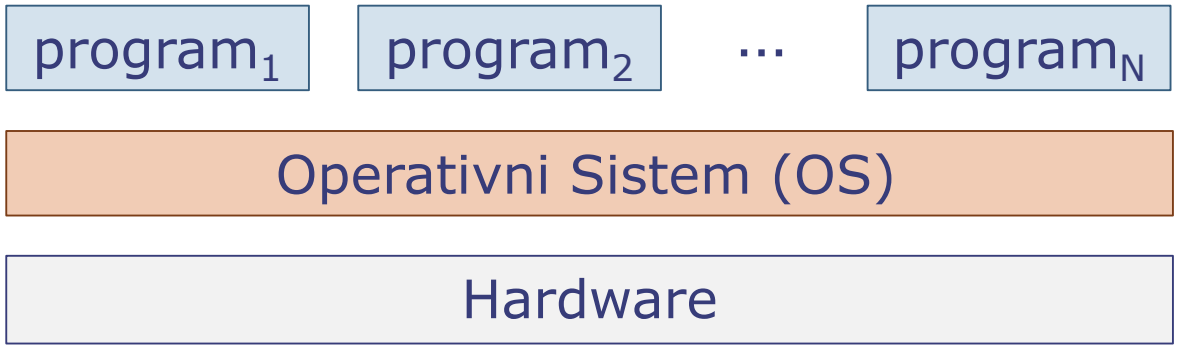
- HW izvrsava jedan program
- Ovaj program ima direktan i kompletan pristup svim svim hardverskim resursima na masini
- Arhitektura (*instruction set architecture* - ISA) je interfejs izmedju HW i SW
- Vecina modernih rac. sistema ne radi ovako!

Operativni sistem



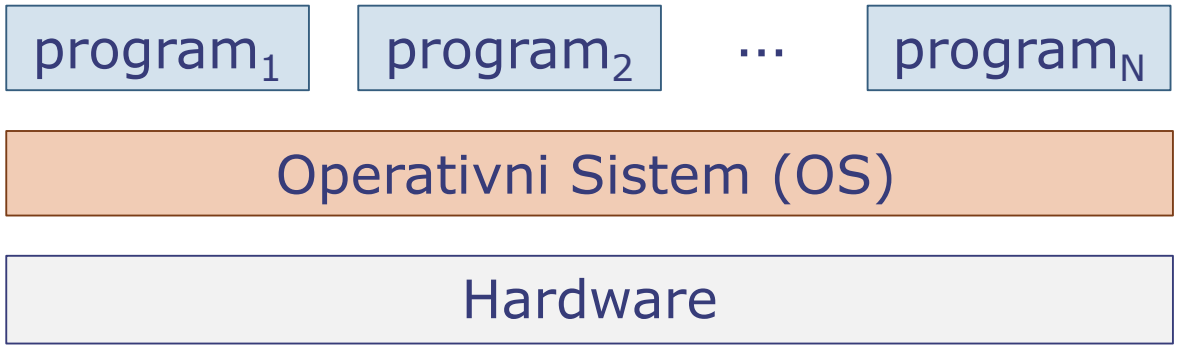
- Vise programa koji se izvrsavaju dele masinu
- Svaki program koji se izvrsava nema direktan pristup hardverskim resursima

Operativni sistem



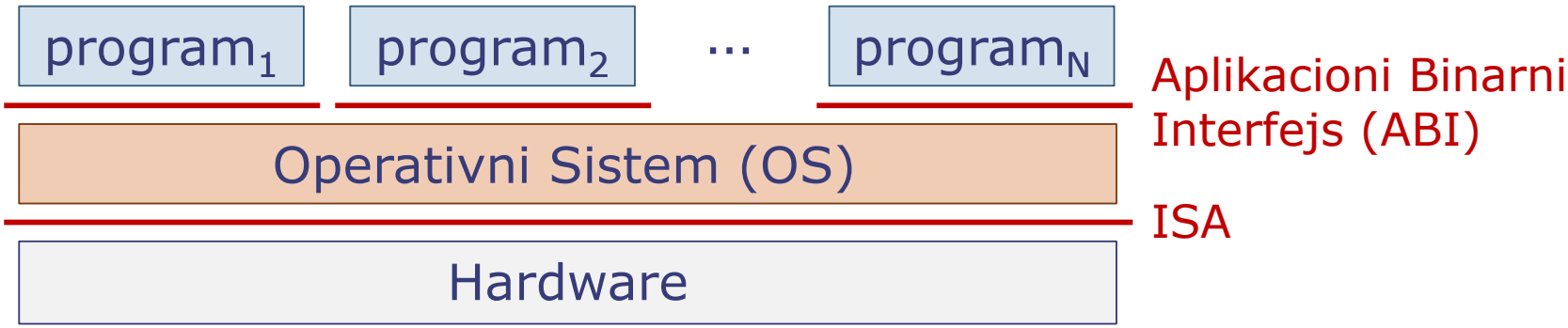
- Vise programa koji se izvrsavaju dele masinu
- Svaki program koji se izvrsava nema direktan pristup hardverskim resursima
- Umesto toga, operativni sistem (OS) kontrolise te programe i nacin na koji oni dele HW resurse

Operativni sistem



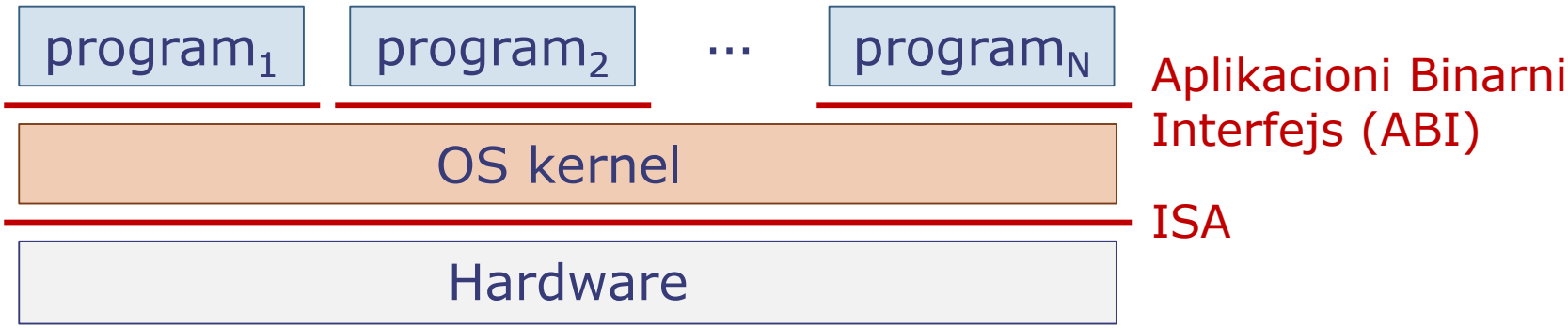
- Vise programa koji se izvrsavaju dele masinu
- Svaki program koji se izvrsava nema direktan pristup hardverskim resursima
- Umesto toga, operativni sistem (OS) kontrolise te programe i nacin na koji oni dele HW resurse
  - Samo OS ima neograniceno pravo pristupa HW

Operativni sistem



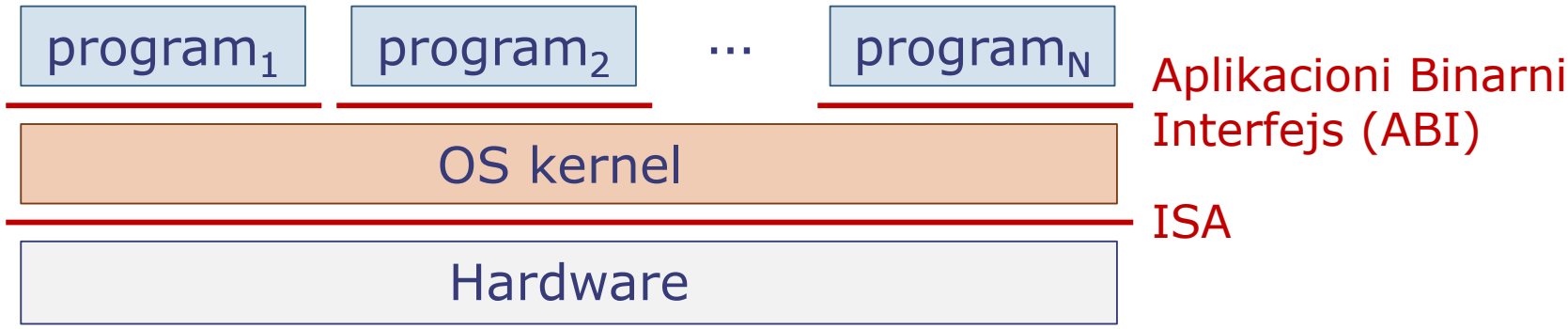
- Vise programa koji se izvrsavaju dele masinu
- Svaki program koji se izvrsava nema direktan pristup hardverskim resursima
- Umesto toga, operativni sistem (OS) kontrolise te programe i nacin na koji oni dele HW resurse
  - Samo OS ima neograniceno pravo pristupa HW
- Aplikacioni binarni interfejs (ABI) je insterfejs izmedju programa i operativnog sistema

Operativni sistem



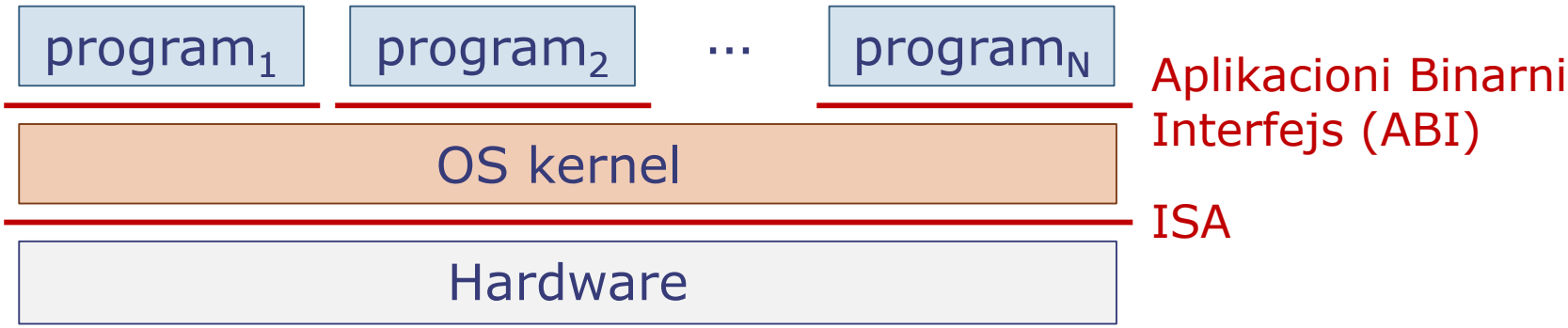
- Program je nista drugo nego niz instrukcija (tj. “samo” koda)

Operativni sistem



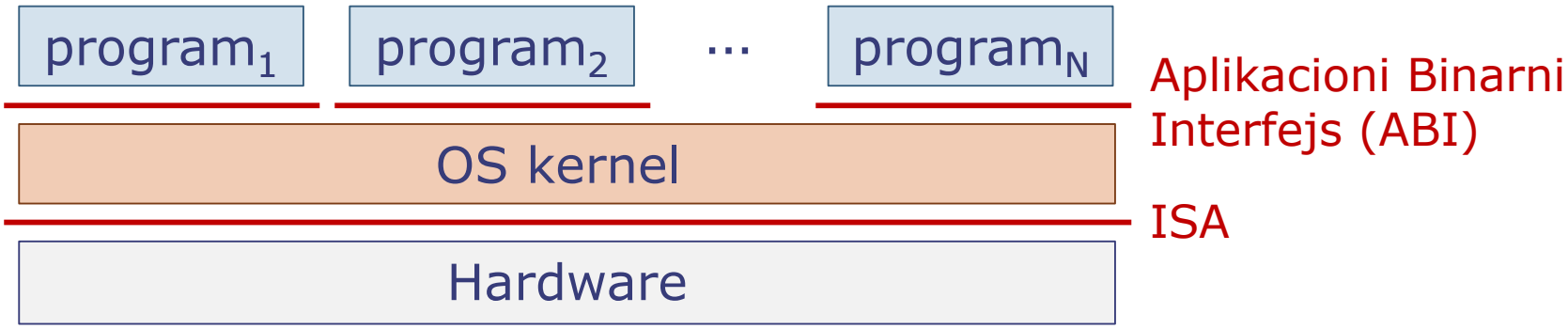
- Program je nista drugo nego niz instrukcija (tj. “samo” koda)
- Proces je instanca programa koji se trenutno izvrsava
  - Ukljucuje kod programa + dodatno stanje (registri, memorija, i drugi resursi

Operativni sistem



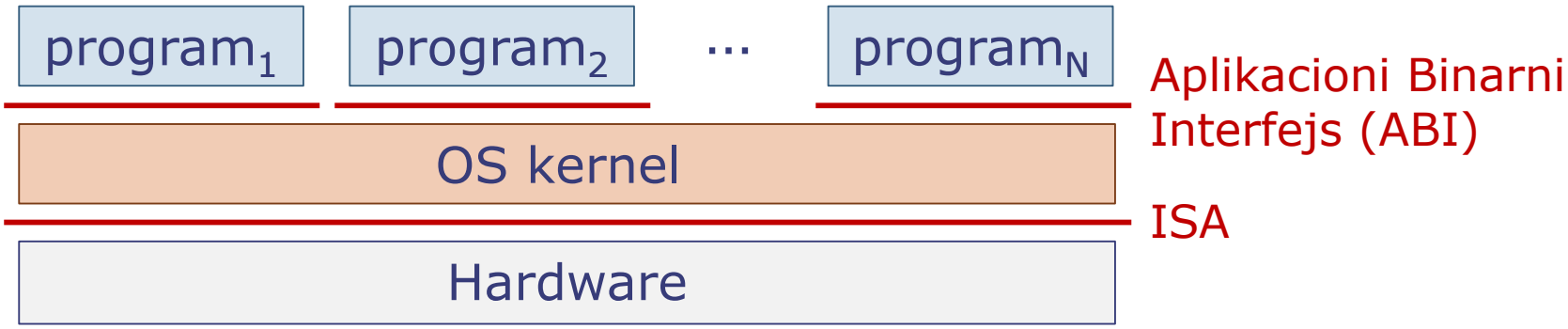
- Program je nista drugo nego niz instrukcija (tj. "samo" koda)
- Proces je instanca programa koji se trenutno izvrsava
  - Ukljucuje kod programa + dodatno stanje (registri, memorija, i drugi resursi
- Kernel OS je takodje proces, ali sa specijalnim privilegijama

Cilj operativnog sistema



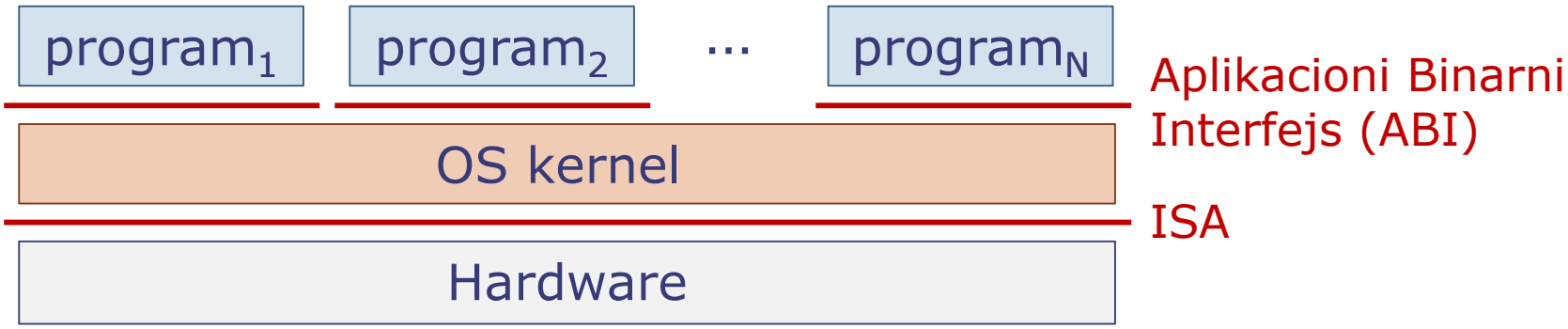
- Zastita i privatnost: Procesi ne mogu pristupati podacima drugih procesa

Cilj operativnog sistema



- Zastita i privatnost: Procesi ne mogu pristupati podacima drugih procesa
- Abstrakcija: OS sakriva detalje vezane za hardverske resurse
  - npr, procesi otvaraju i pristupaju datotekama umesto slanja direktnih instrukcija periferijskim uredjajima, pa cak i disku

Cilj operativnog sistema



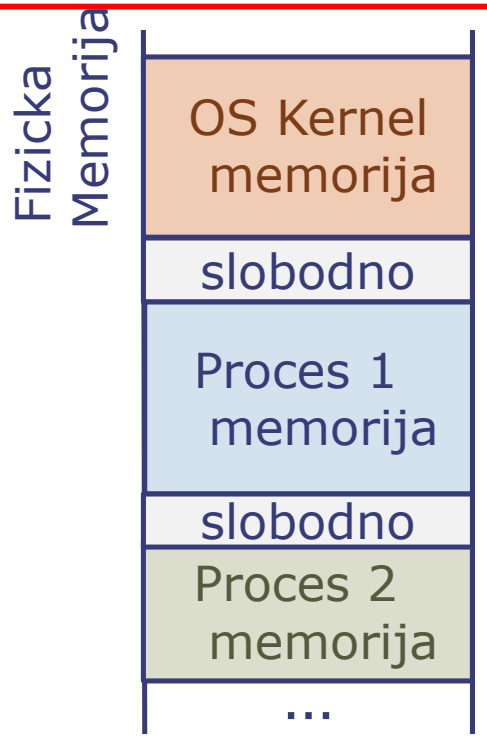
- Zastita i privatnost: Procesi ne mogu pristupati podacima drugih procesa
- Abstrakcija: OS sakriva detalje vezane za hardverske resurse
  - npr, procesi otvaraju i pristupaju datotekama umesto slanja direktnih instrukcija periferijskim uredjajima, pa cak i disku
- Menadzment resursa: OS kontrolise kako procesi dele resurse (CPU, memorija, disk, itd.)

Operativni sistemi: Sira slika

- OS kernel obezbedjuje privatni adresni prostor svakom procesu
  - Svaki proces alocira prostor u fizickoj memoriji koriscenjem OS
  - Proces ne moze da pristupa memoriji dodeljenoj drugom procesu

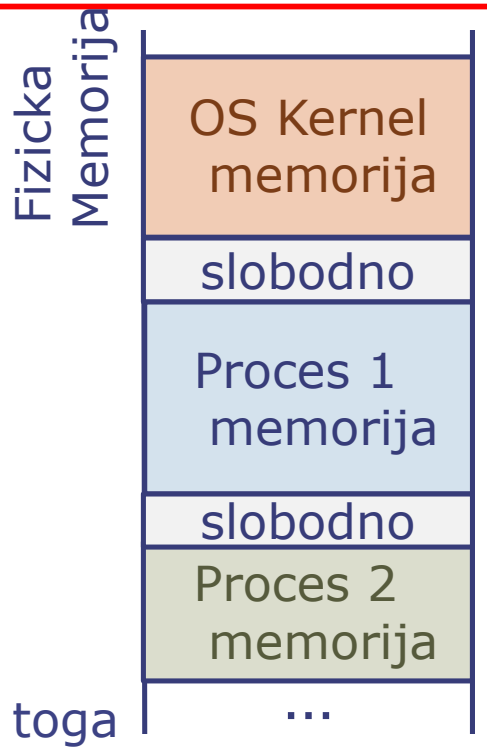
Operativni sistemi: Sira slika

- OS kernel obezbedjuje privatni adresni prostor svakom procesu
  - Svaki proces alocira prostor u fizickoj memoriji koriscenjem OS
  - Proces ne moze da pristupa memoriji dodeljenoj drugom procesu



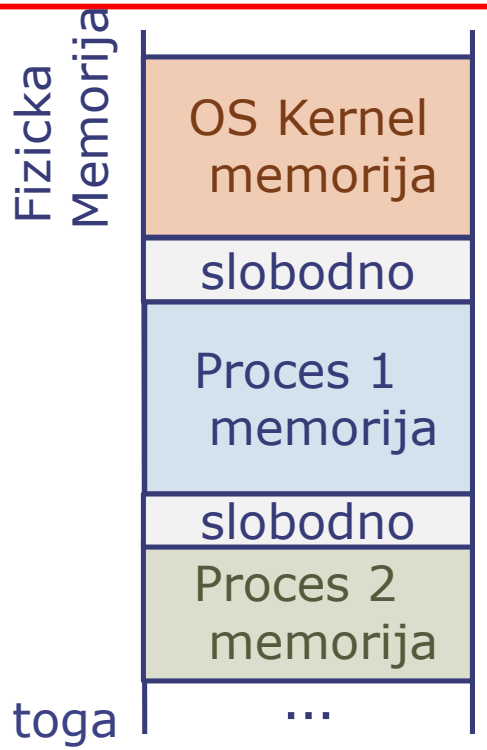
Operativni sistemi: Sira slika

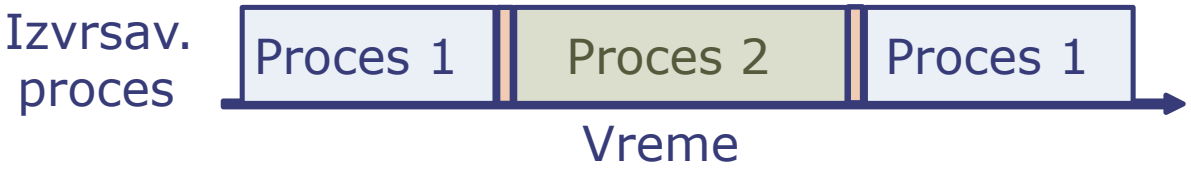
- OS kernel obezbedjuje privatni adresni prostor svakom procesu
  - Svaki proces alocira prostor u fizickoj memoriji koriscenjem OS
  - Proces ne moze da pristupa memoriji dodeljenoj drugom procesu
- OS kernel rasporedjuje procese za izvrsavanje na CPU
  - Svaki proces dobija kolicinu CPU vremena
  - Proces ne moze koristiti vise CPU vremena od toga



Operativni sistemi: Sira slika

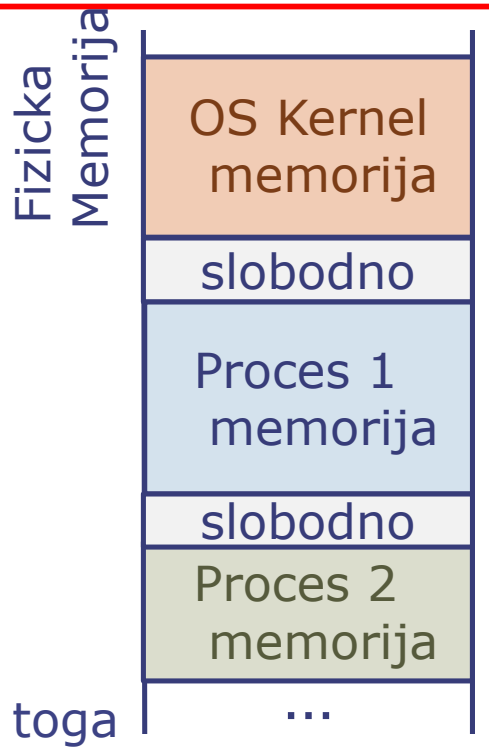
- OS kernel obezbedjuje privatni adresni prostor svakom procesu
  - Svaki proces alocira prostor u fizickoj memoriji koriscenjem OS
  - Proces ne moze da pristupa memoriji dodeljenoj drugom procesu
- OS kernel rasporedjuje procese za izvrsavanje na CPU
  - Svaki proces dobija kolicinu CPU vremena
  - Proces ne moze koristiti vise CPU vremena od toga

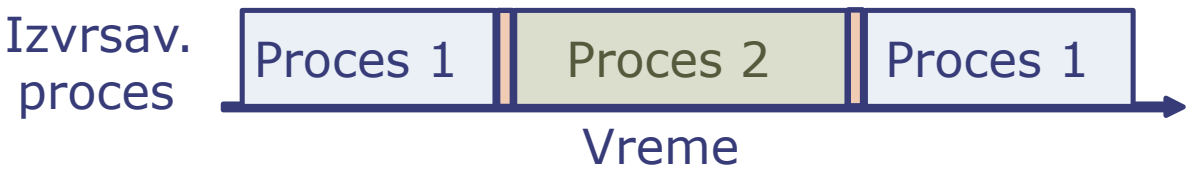




Operativni sistemi: Sira slika

- OS kernel obezbedjuje privatni adresni prostor svakom procesu
  - Svaki proces alocira prostor u fizickoj memoriji koriscenjem OS
  - Proces ne moze da pristupa memoriji dodeljenoj drugom procesu
- OS kernel rasporedjuje procese za izvrsavanje na CPU
  - Svaki proces dobija kolicinu CPU vremena
  - Proces ne moze koristiti vise CPU vremena od toga





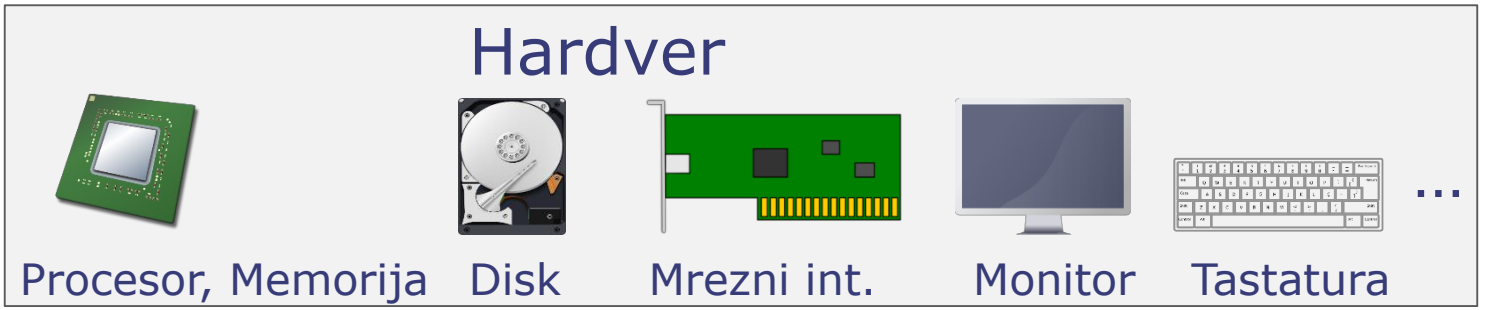
- OS kernel dozvoljava procesima poziv sistemskih servisa (npr. pristup datotekama ili mreznim soketima) putem sistemskih poziva

Virtualna Masina
Novi sloj apstrakcije

- OS kreira Virtualnu Masinu (VM) za svaki proces
  - Svaki proces veruje da se izvrsava na sopstvenoj masini…
  - …ali ta masina ne postoji u fizickom hardveru

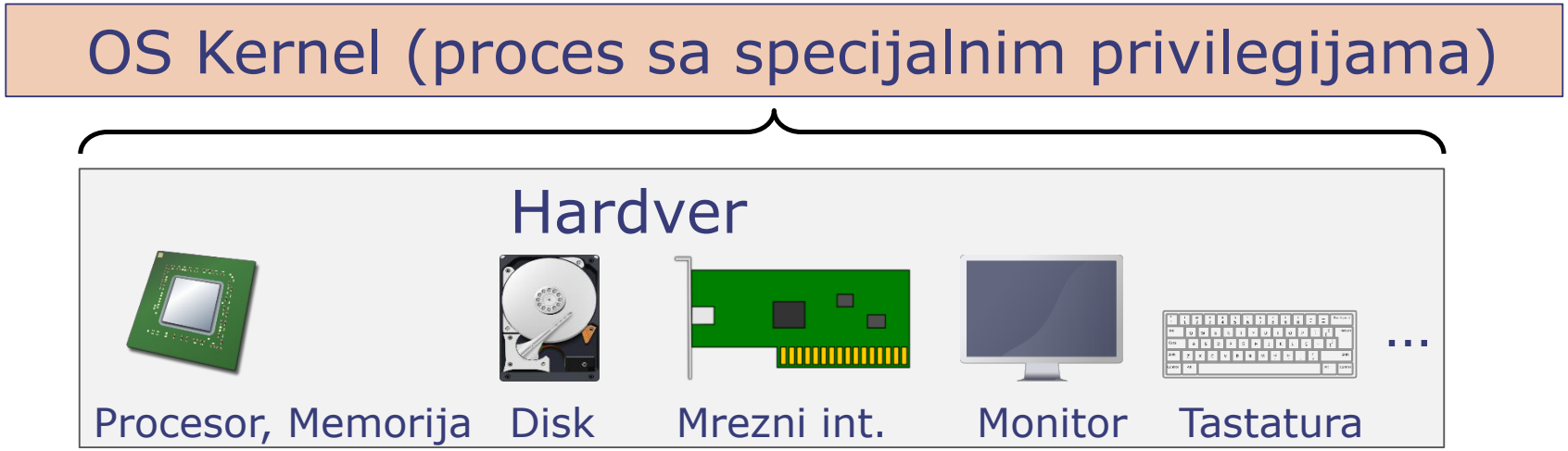
Virtualna Masina
Novi sloj apstrakcije

- OS kreira Virtualnu Masinu (VM) za svaki proces
  - Svaki proces veruje da se izvrsava na sopstvenoj masini…
  - …ali ta masina ne postoji u fizickom hardveru



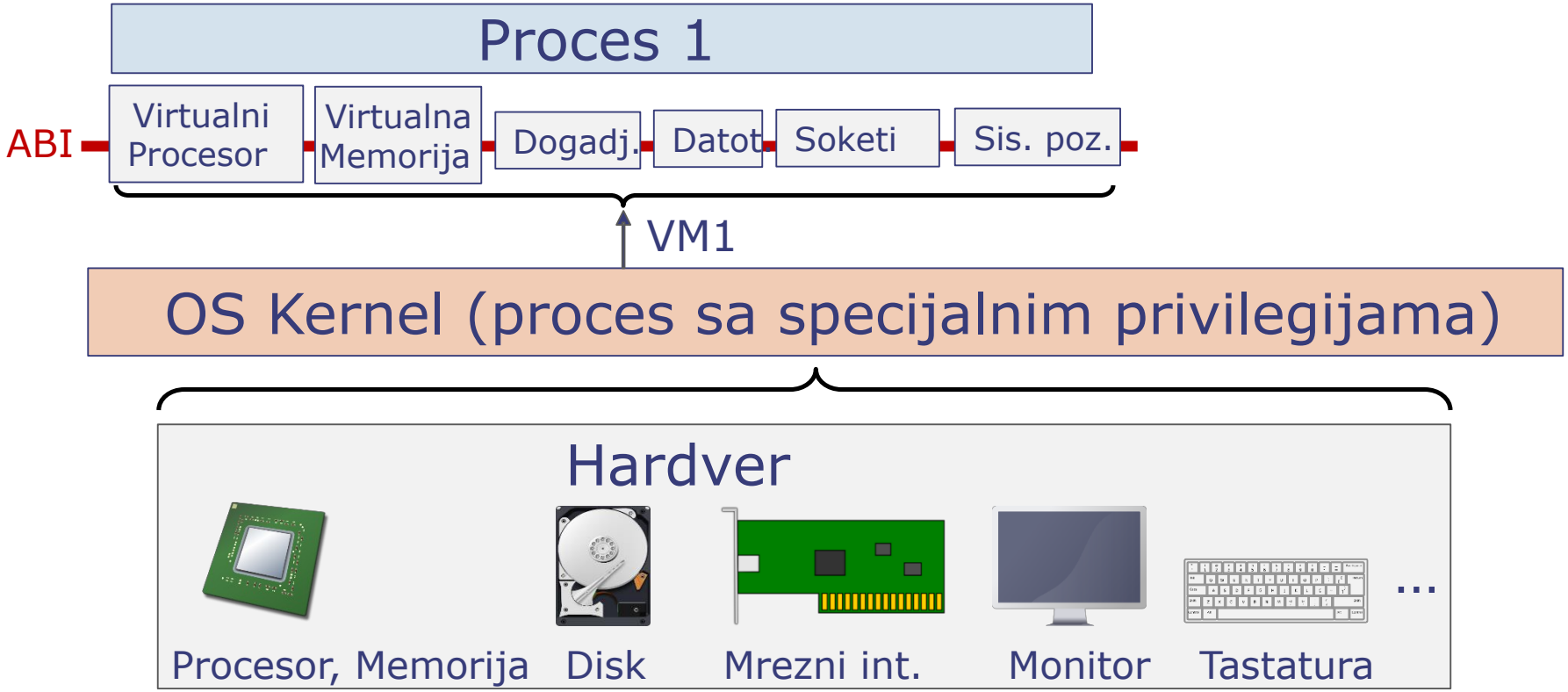
Virtualna Masina
Novi sloj apstrakcije

- OS kreira Virtualnu Masinu (VM) za svaki proces
  - Svaki proces veruje da se izvrsava na sopstvenoj masini…
  - …ali ta masina ne postoji u fizickom hardveru



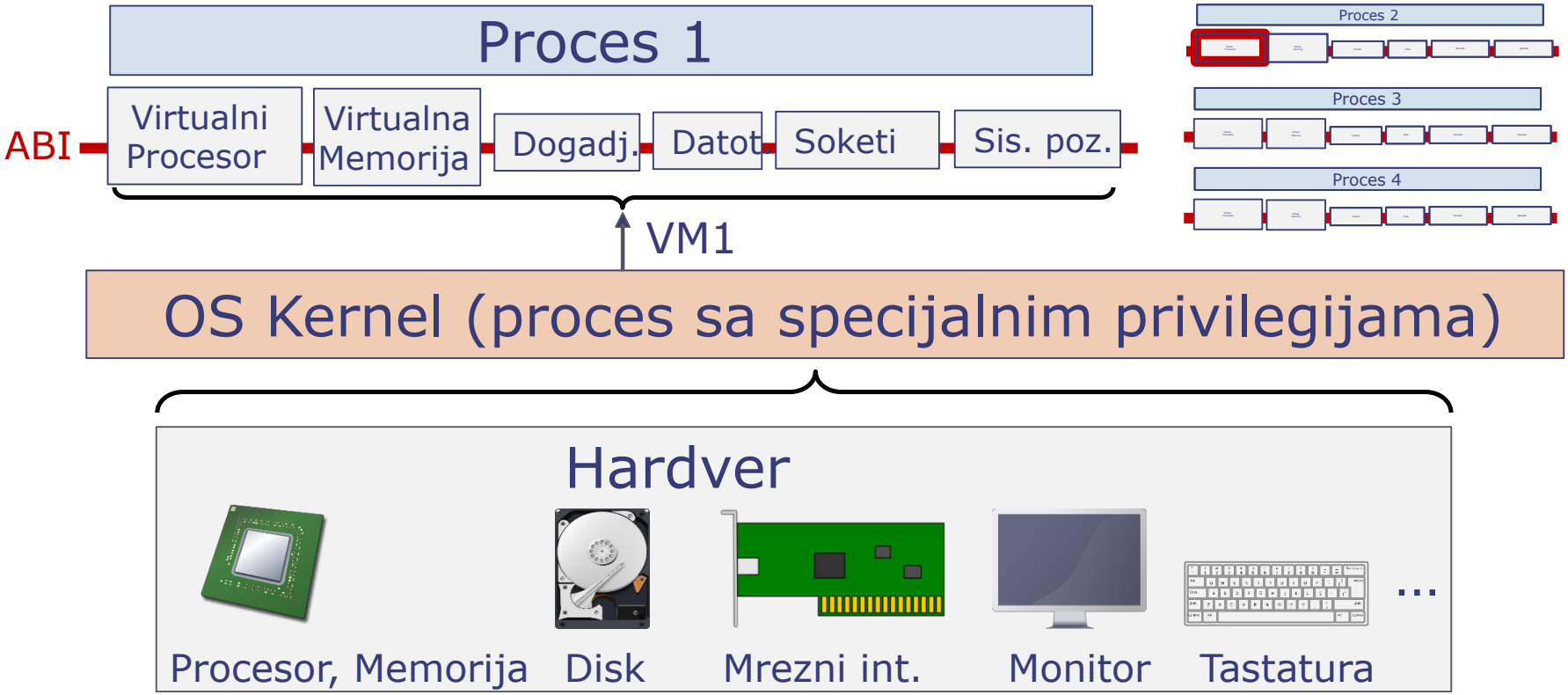
Virtualna Masina
Novi sloj apstrakcije

- OS kreira Virtualnu Masinu (VM) za svaki proces
  - Svaki proces veruje da se izvrsava na sopstvenoj masini…
  - …ali ta masina ne postoji u fizickom hardveru



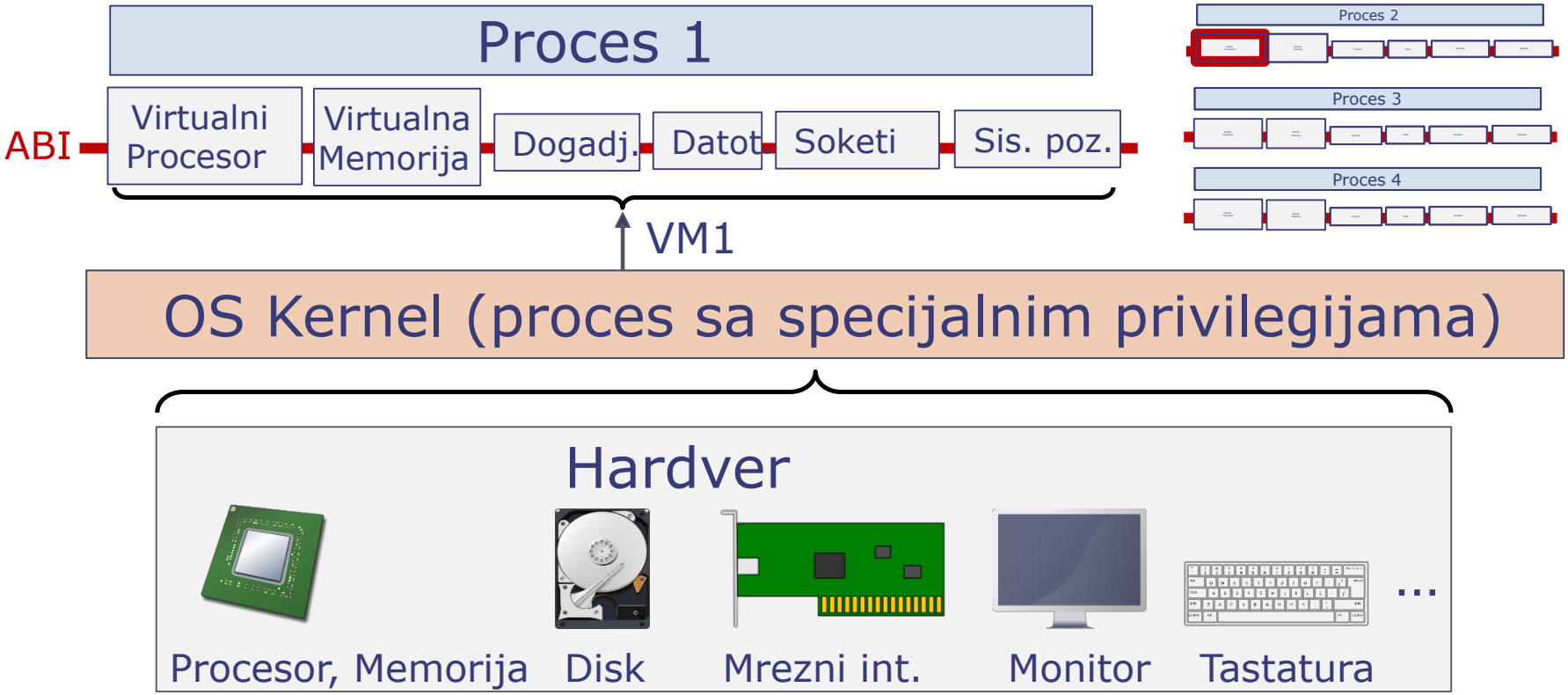
Virtualna Masina
Novi sloj apstrakcije

- OS kreira Virtualnu Masinu (VM) za svaki proces
  - Svaki proces veruje da se izvrsava na sopstvenoj masini…
  - …ali ta masina ne postoji u fizickom hardveru



Virtualna Masina
Novi sloj apstrakcije

- VM je emulacija racunarskog sistema
  - Veoma generalan koncept, daleko izvan OS



Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?

Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?

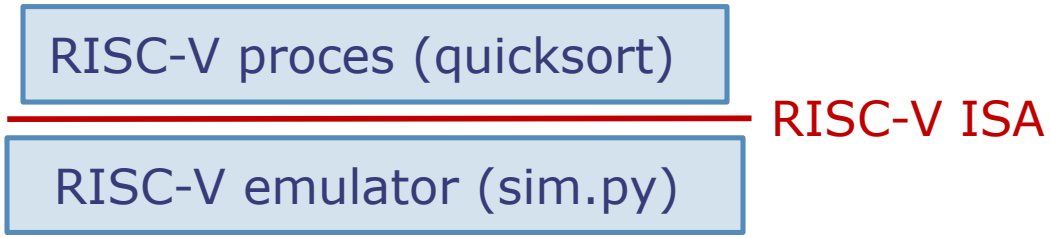
RISC-V proces (quicksort)

Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?

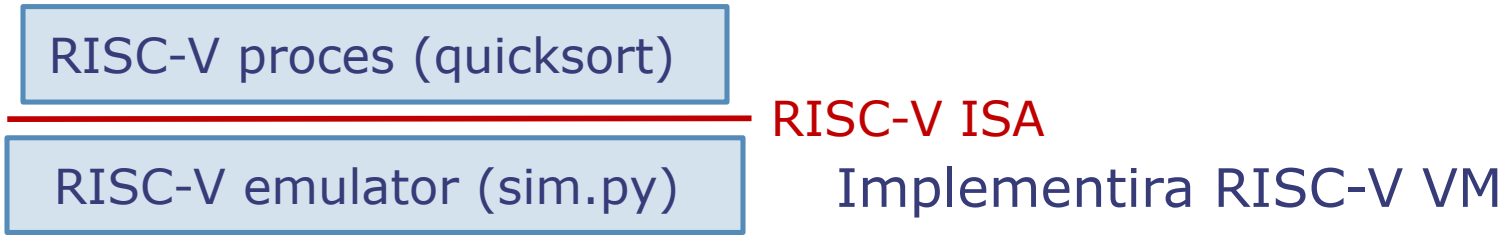
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



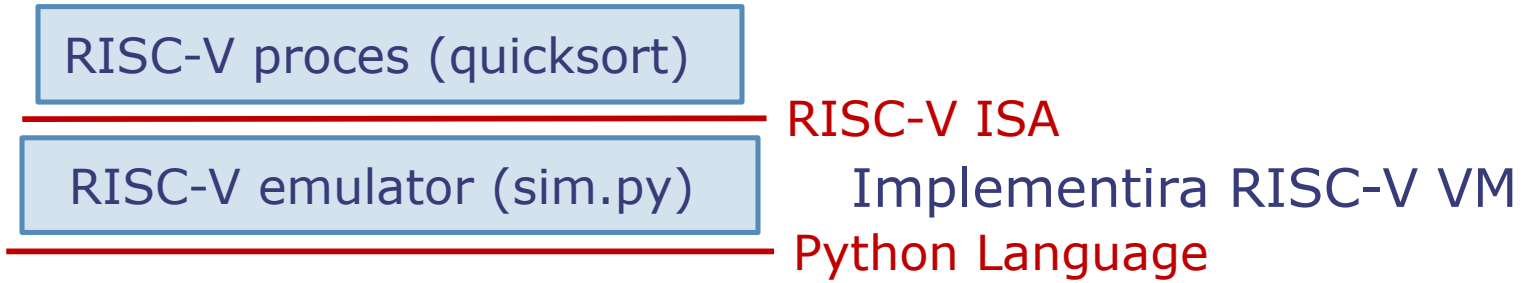
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



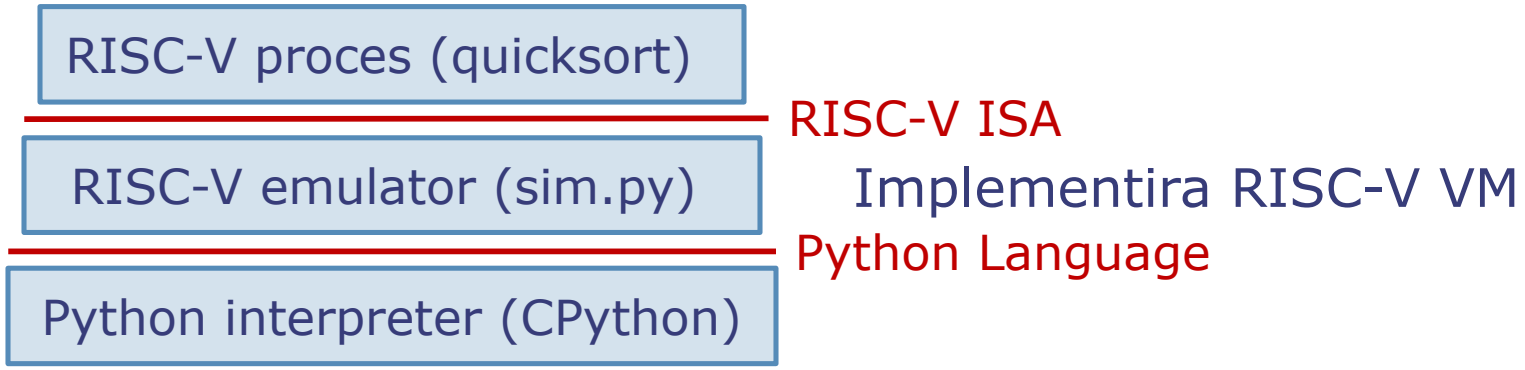
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



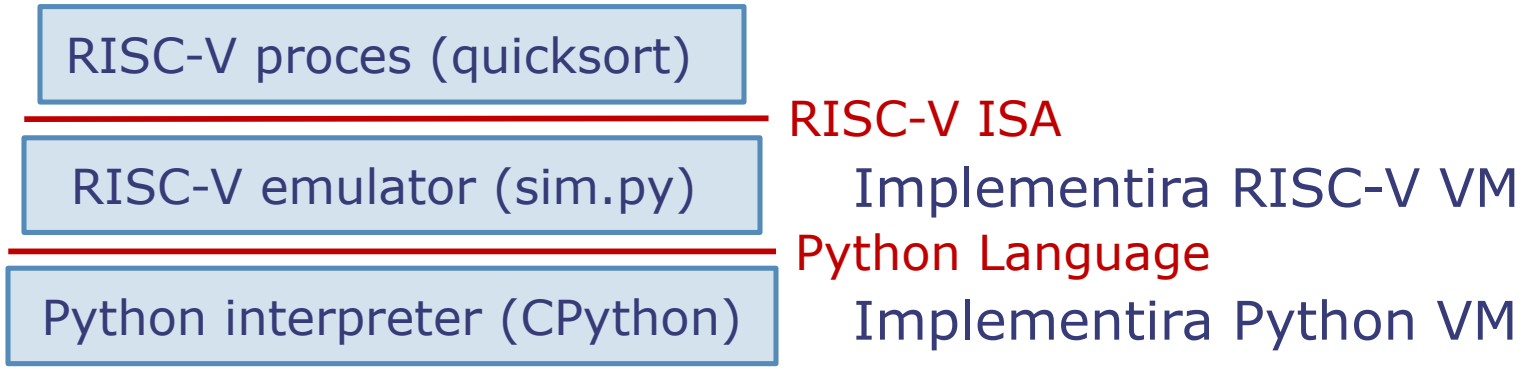
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



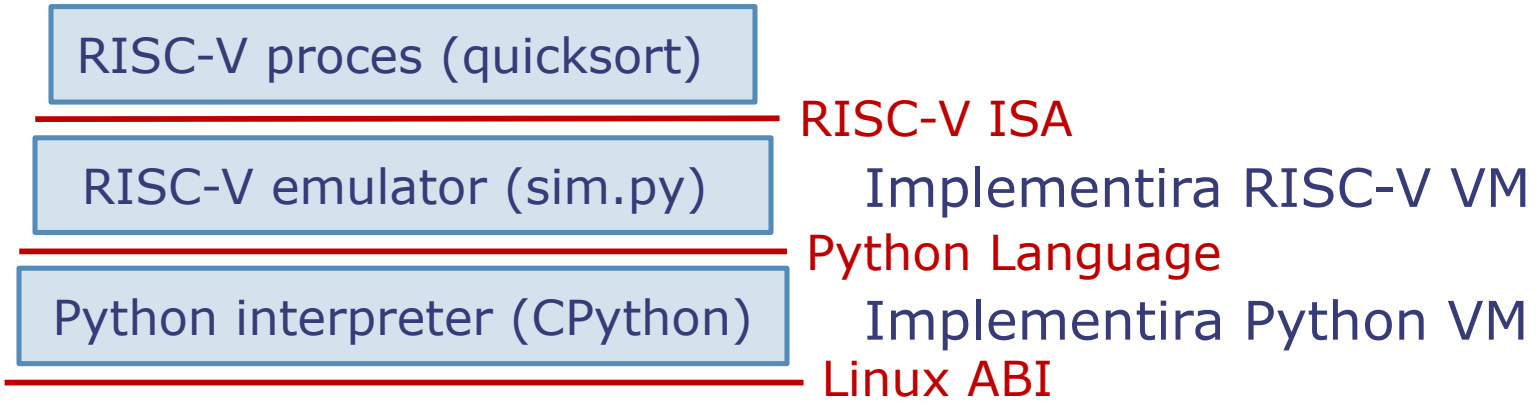
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



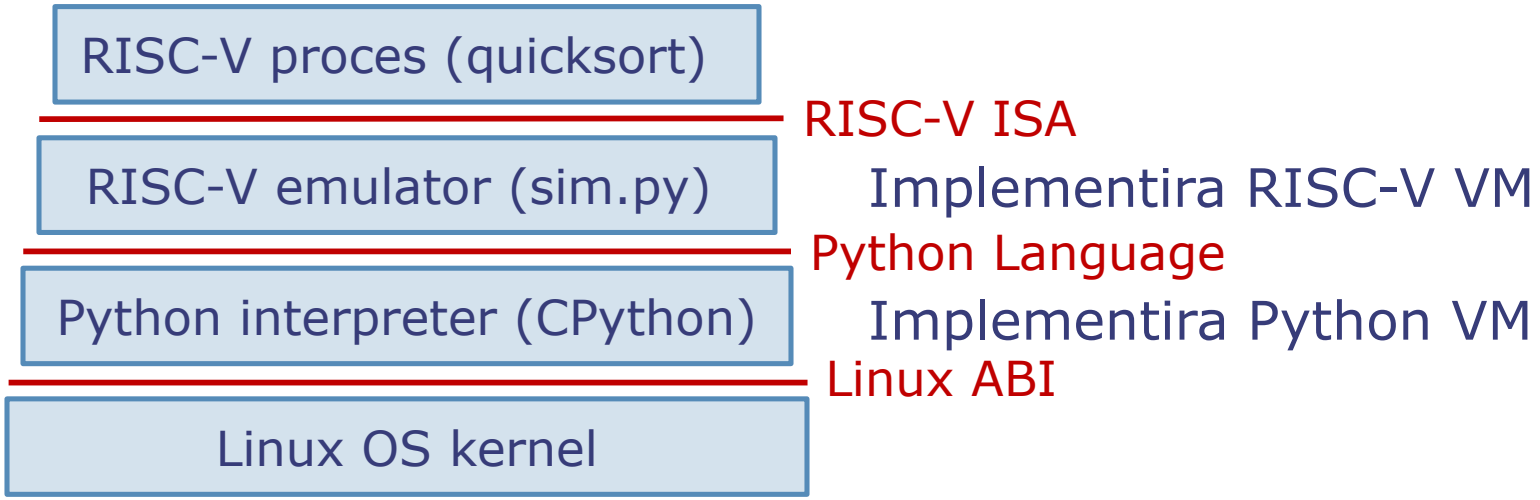
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



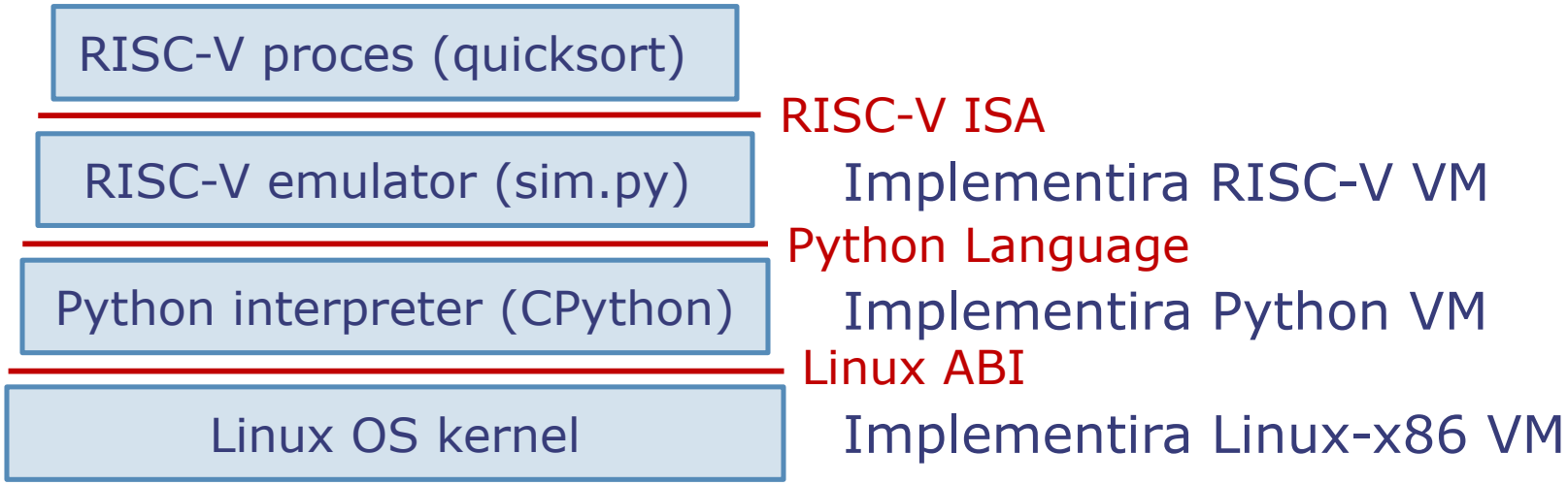
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



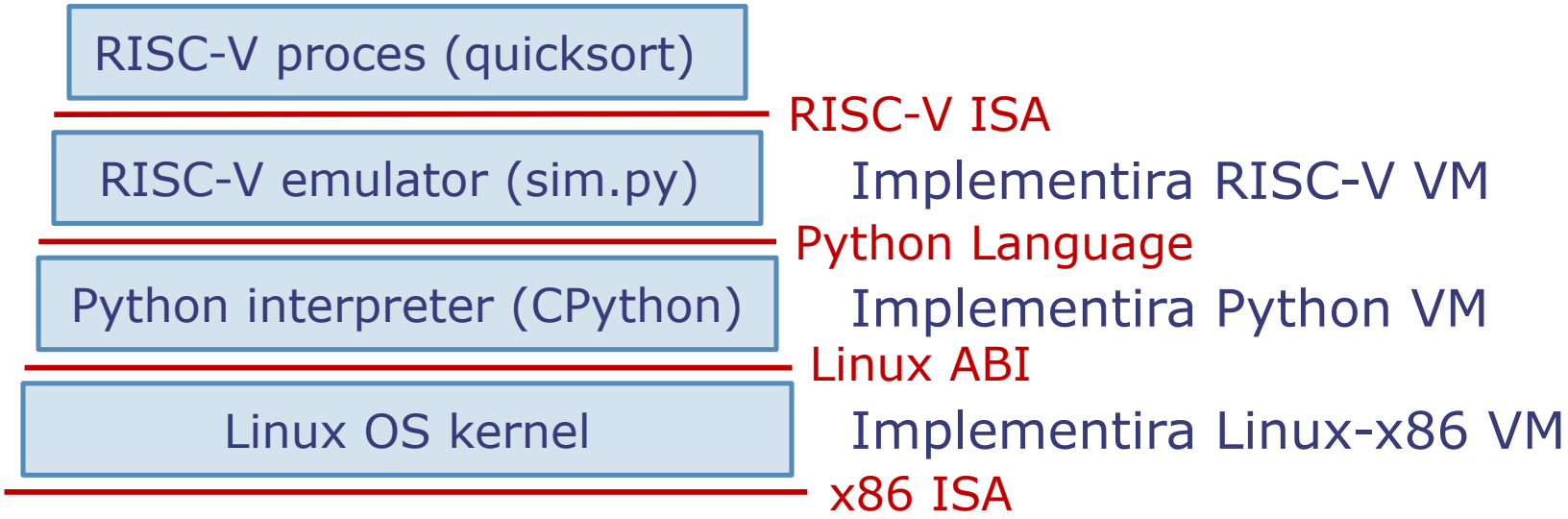
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



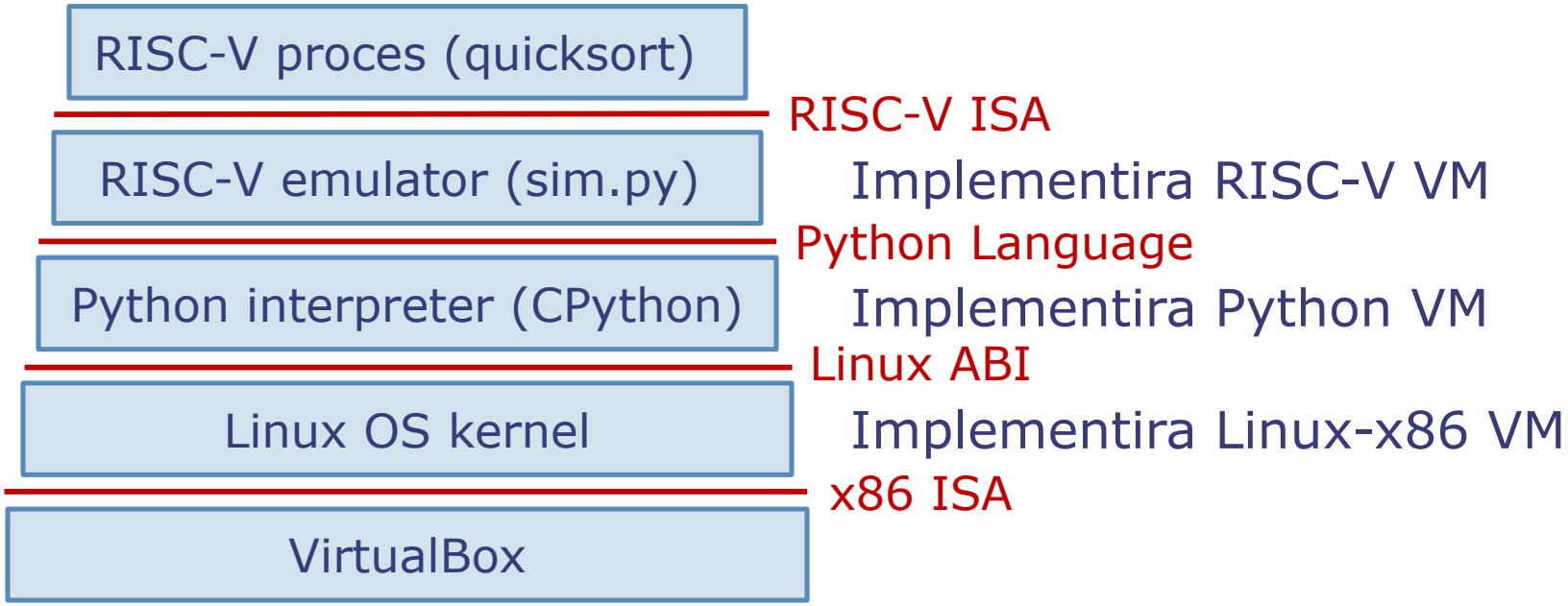
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



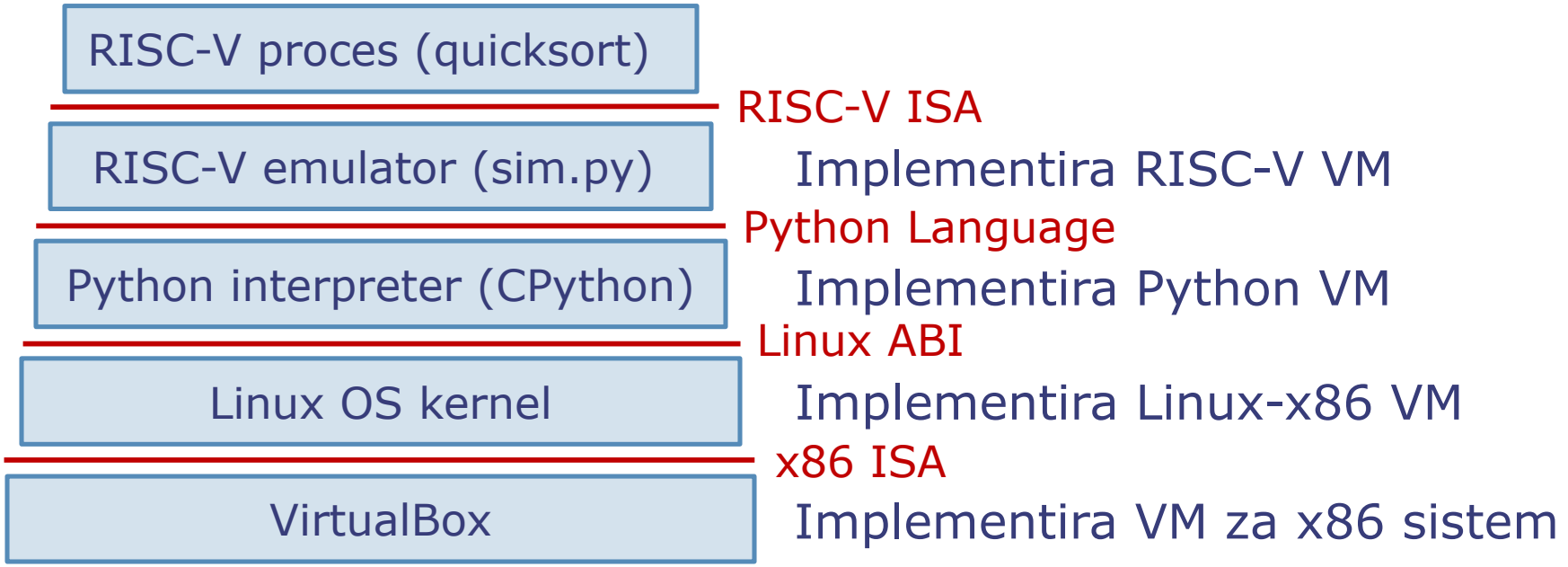
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



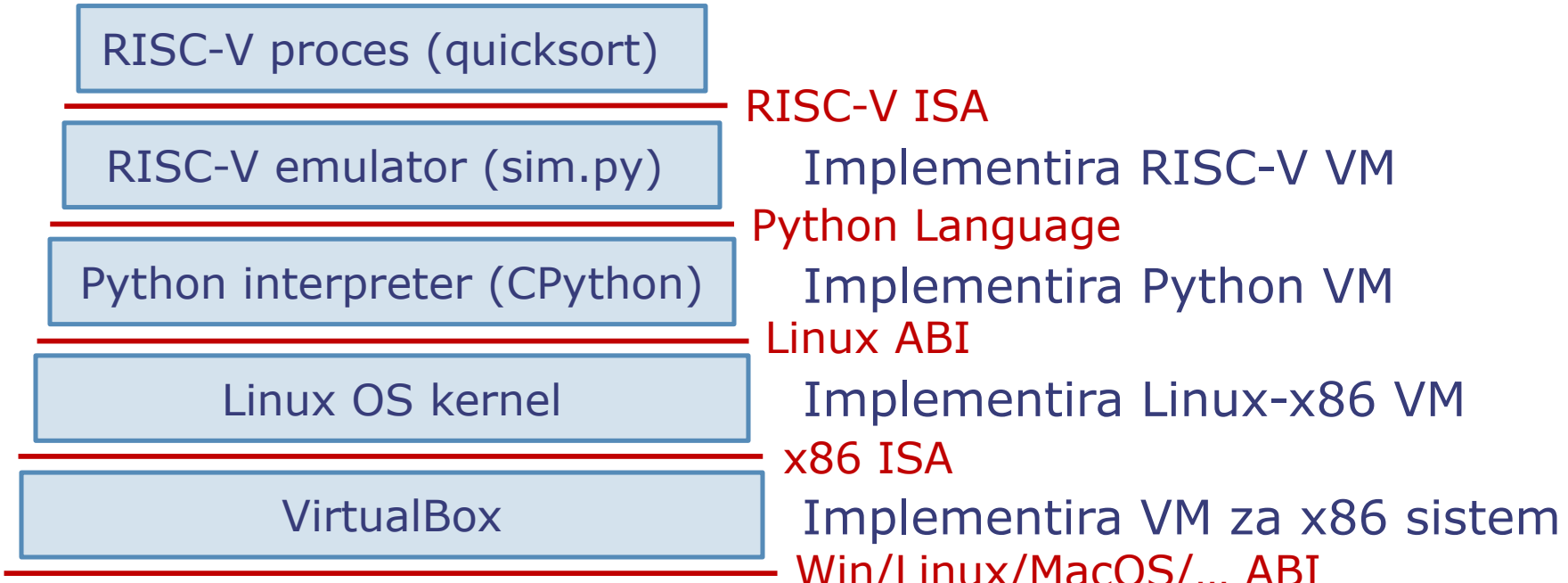
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



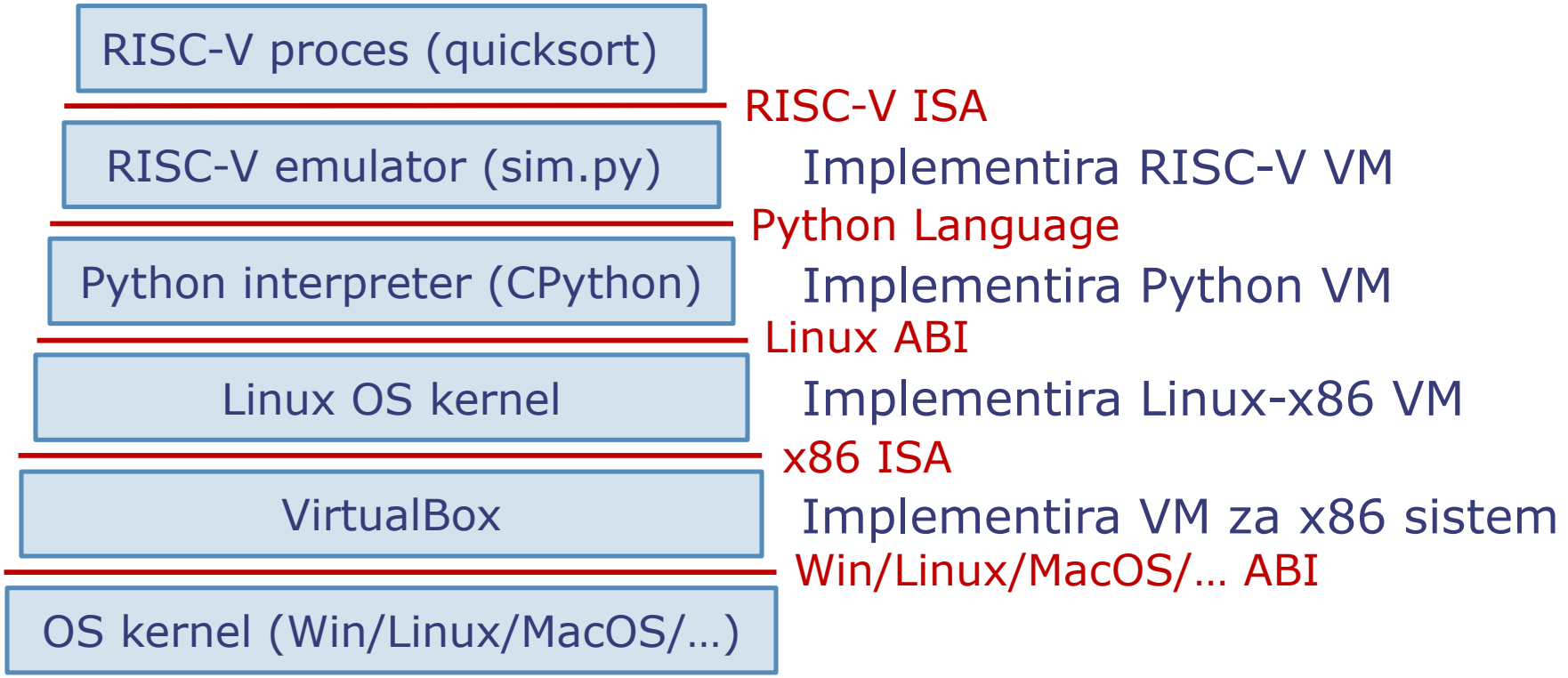
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



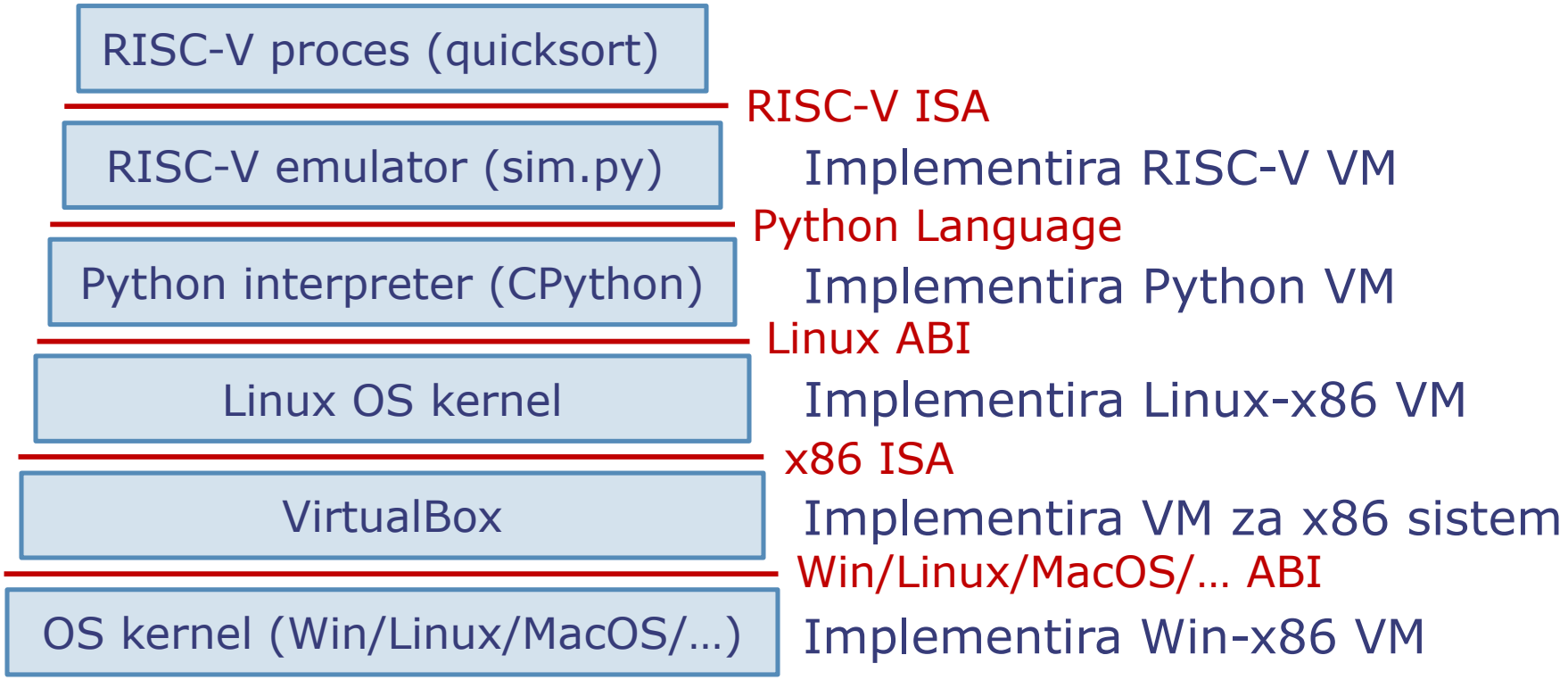
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



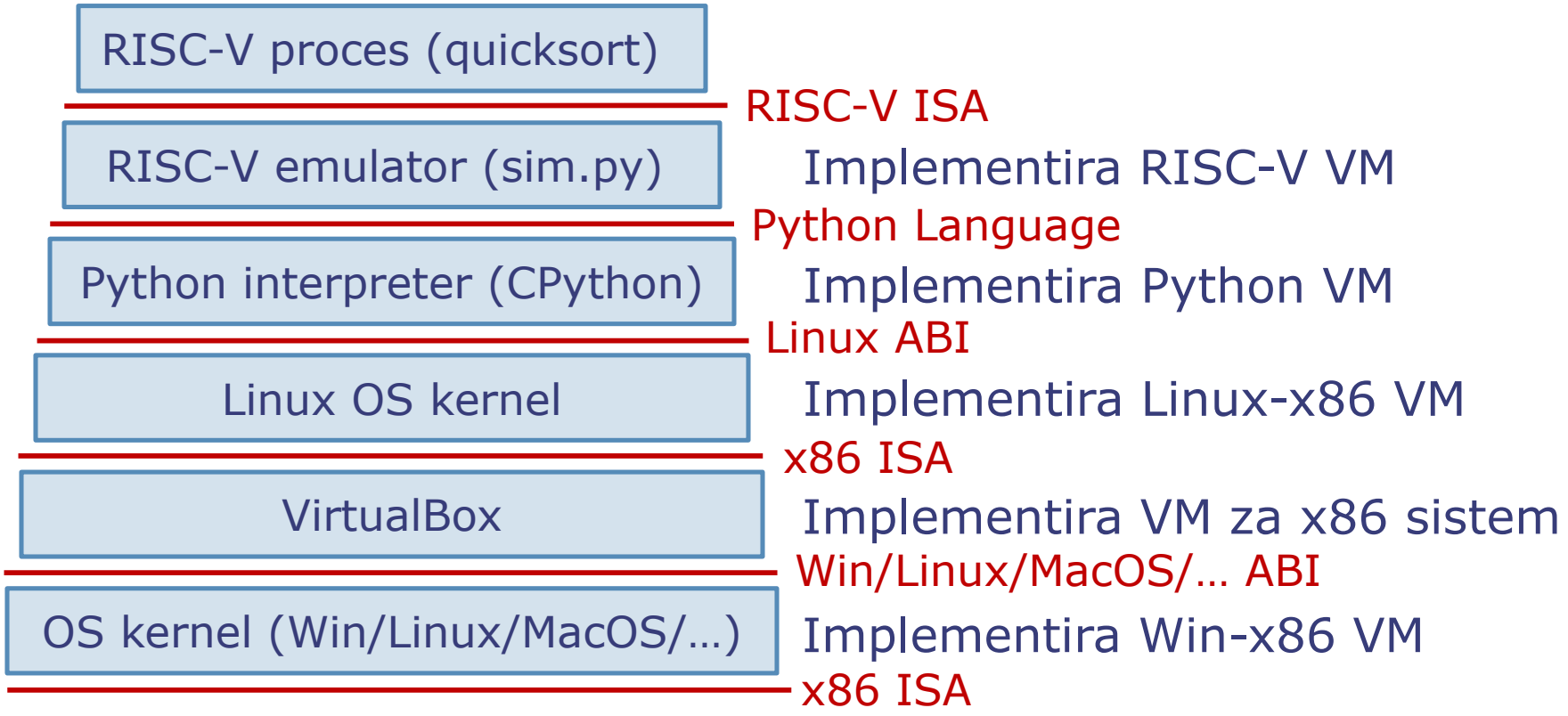
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



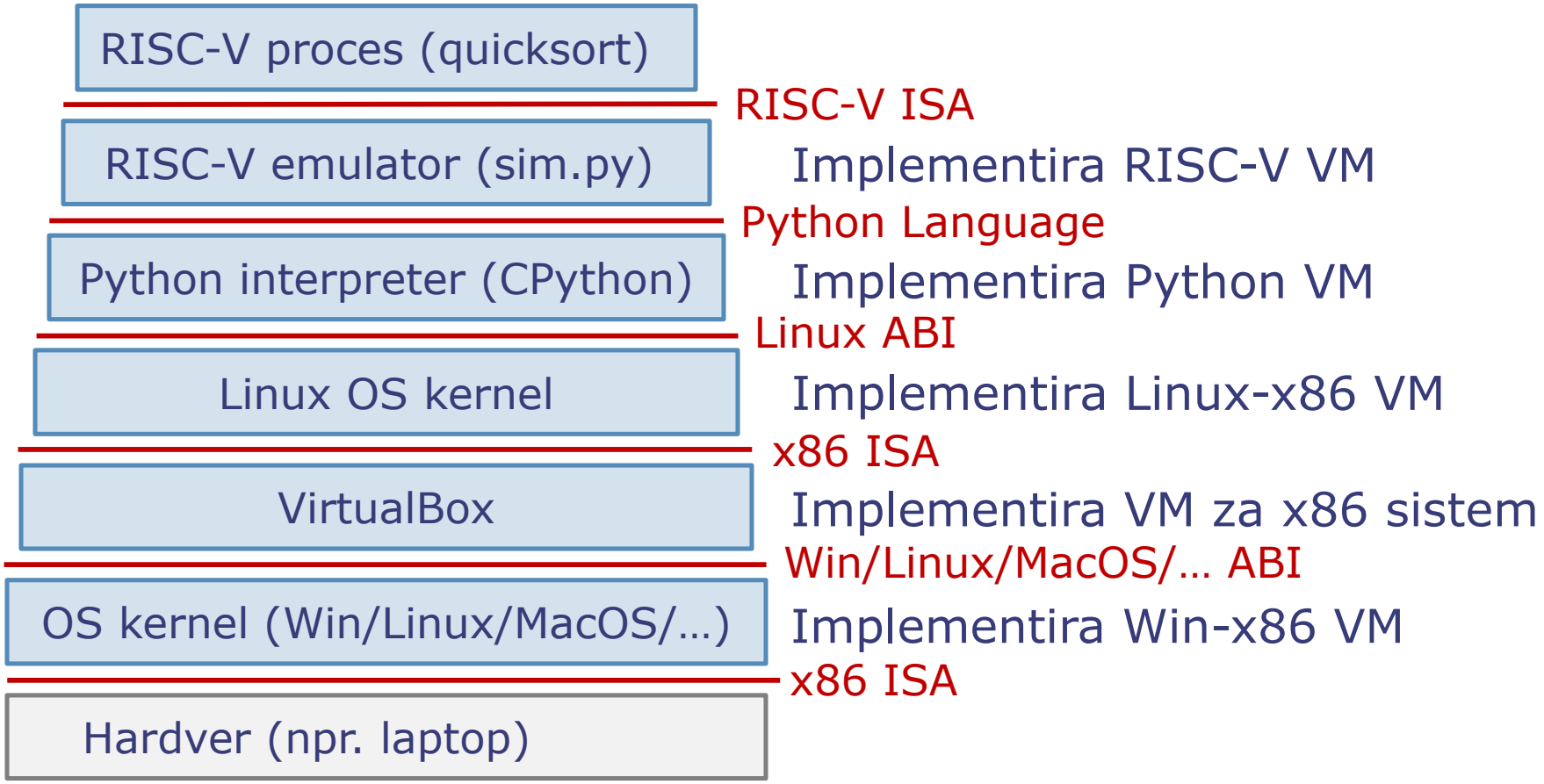
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



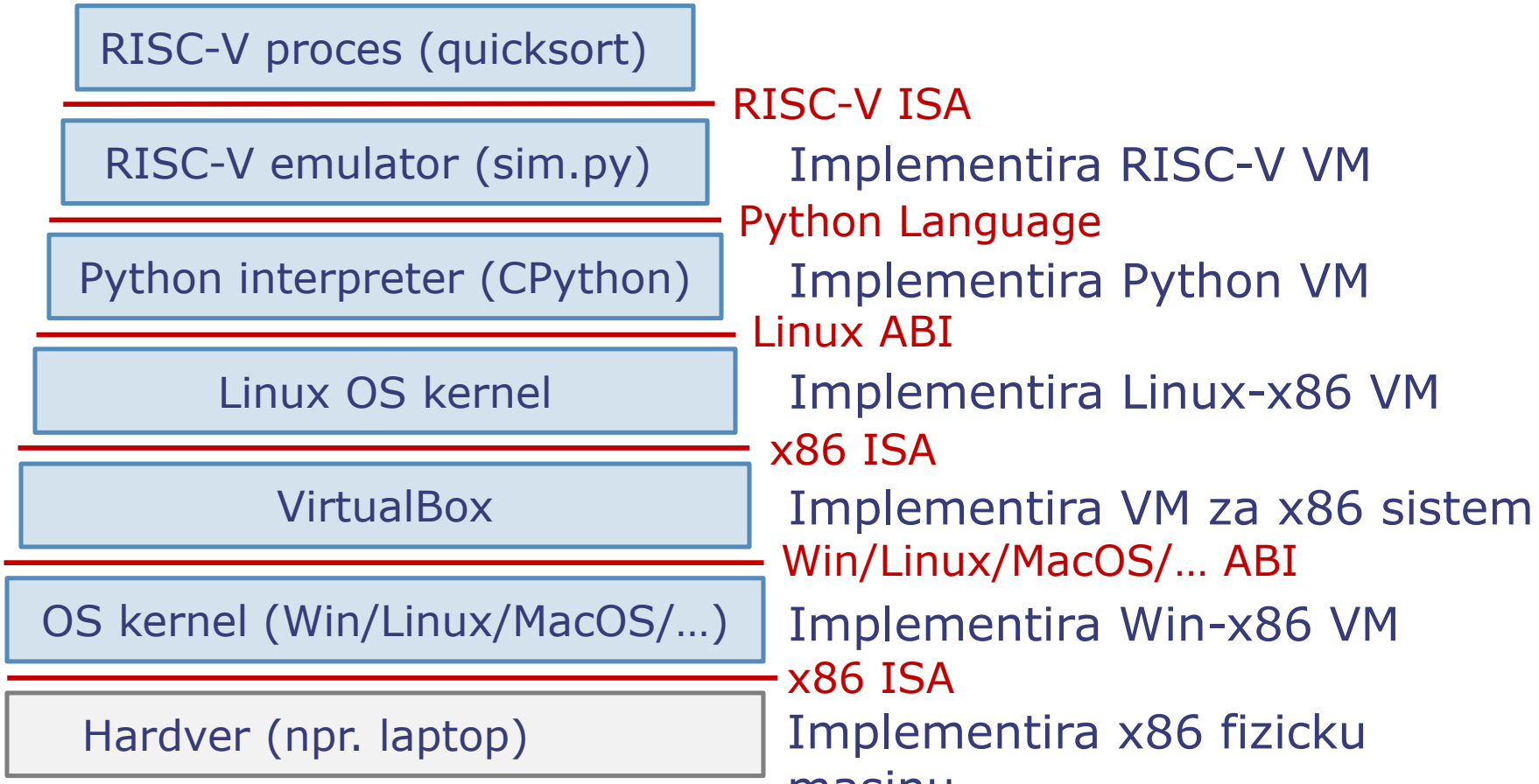
Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



Virtualne Masine Svuda

- Primer: Koliko VM se ovde koristi?



Implementacija Virtualne Masine

- Virtualne masine mogu biti implementirane potpuno u softveru, ali po cenu drasticno losijih performansi
  - npr Python programi su 10-100x sporiji nego “native” Linux programi usled dodatnog procesiranja Python interpretera

Implementacija Virtualne Masine

- Virtualne masine mogu biti implementirane potpuno u softveru, ali po cenu drasticno losijih performansi
  - npr Python programi su 10-100x sporiji nego “native” Linux programi usled dodatnog procesiranja Python interpretera
- Zelimo podrsku operativnog sistema sa minimalnim dodatnim procesir. → hardverska podrska za virtualne masine!

ISA Dodatak kao podrska OS

ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu

ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu
- Privilegovane instrukcije and registri su jedino dostupni u nadziranom modu

ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu
- Privilegovane instrukcije and registri su jedino dostupni u nadziranom modu
- Prekidi i izuzeci obezbedjuju bezbedne tranzicije iz korisnickog u nadzirani mod

ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu
- Privilegovane instrukcije and registri su jedino dostupni u nadziranom modu
- Prekidi i izuzeci obezbedjuju bezbedne tranzicije iz korisnickog u nadzirani mod
- Virtualna memorija obezbedjuje privatni adresni prostor i apstakuje smestajne resurse na masini

ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu
- Privilegovane instrukcije and registri su jedino dostupni u nadziranom modu
- Prekidi i izuzeci obezbedjuju bezbedne tranzicije iz korisnickog u nadzirani mod
- Virtualna memorija obezbedjuje privatni adresni prostor i apstakuje smestajne resurse na masini

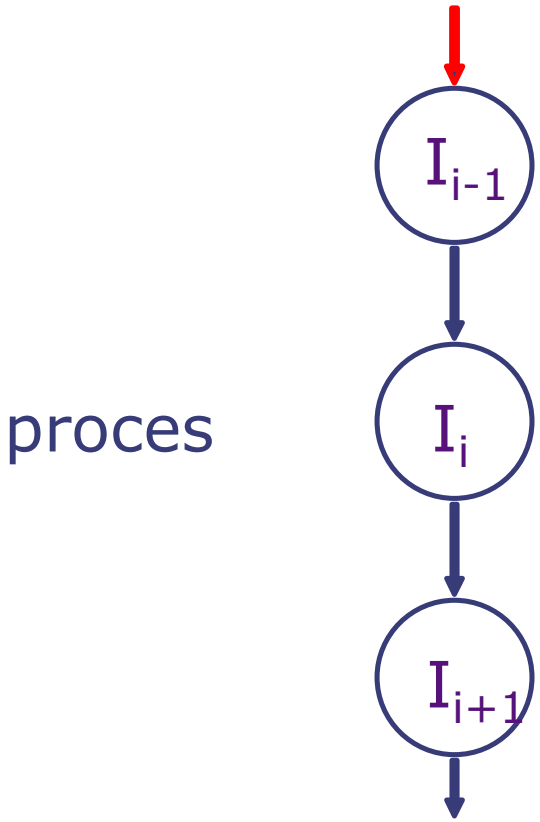
ISA Dodatak kao podrska OS

- Dva moda izvrsavanja: korisnicki i nadzirani
  - OS kernel se izvrsava u nadziranom (supervisor) modu
  - Svi ostali procesi se izvrsavaju u korisnickom (user) modu
- Privilegovane instrukcije and registri su jedino dostupni u nadziranom modu
- Prekidi i izuzeci obezbedjuju bezbedne tranzicije iz korisnickog u nadzirani mod
- Virtualna memorija obezbedjuje privatni adresni prostor i apstakuje smestajne resurse na masini

Ovi ISA dodaci funkcionisu samo ako se hardver i softver (OS) dogovore oko zajednickog skupa konvencija!

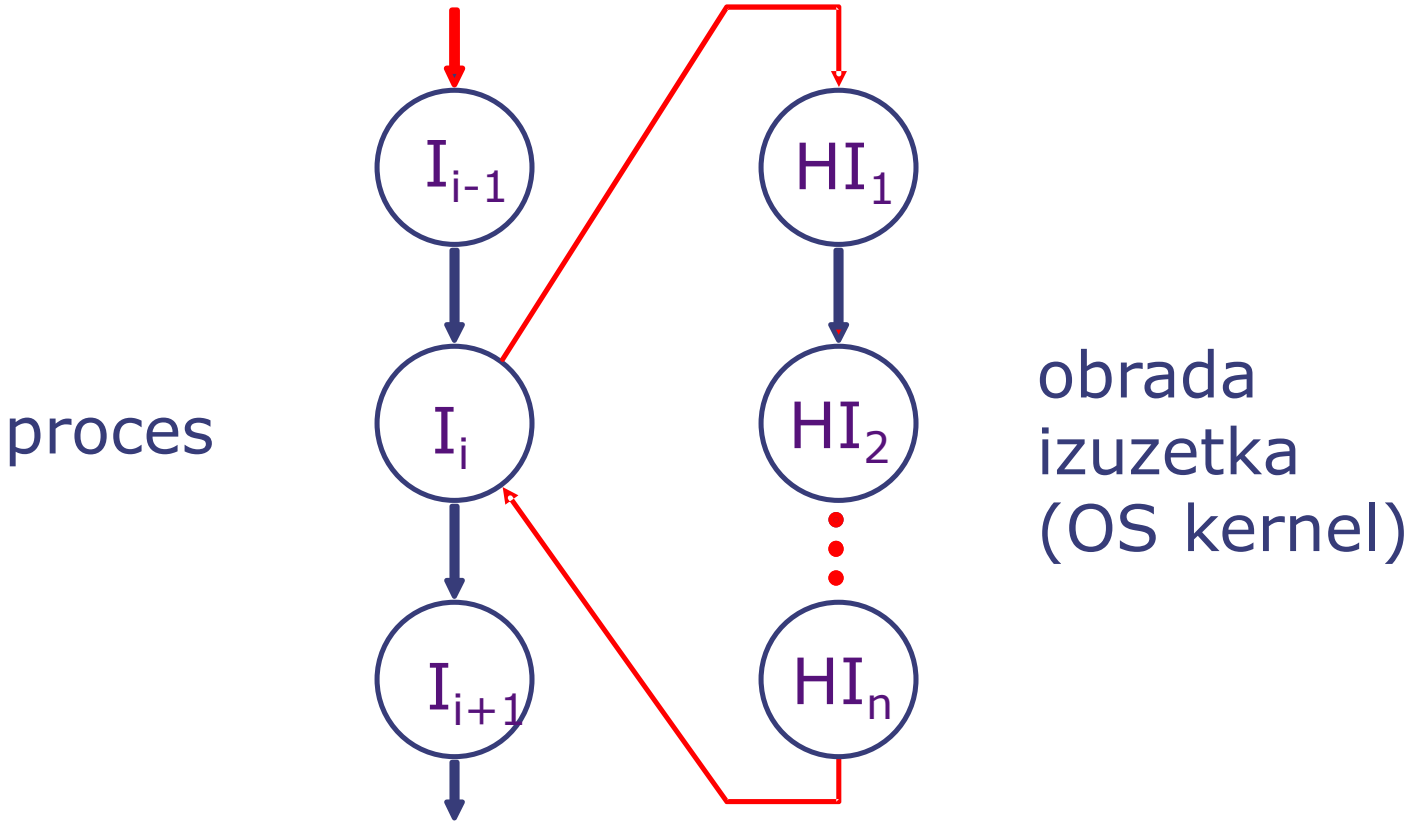
Izuzeci

- Izuzetak je dogadjaj koji mora da se procesira unutar OS kernela. On je obicno neocekivan i redak.



Izuzeci

- Izuzetak je dogadjaj koji mora da se procesira unutar OS kernela. On je obicno neocekivan i redak.



Uzroci izuzetaka

- Pojmovi izuzetak i prekid se uglavnom koriste zamenjujuci jedan drugi, uz male razlike:

Uzroci izuzetaka

- Pojmovi izuzetak i prekid se uglavnom koriste zamenjujuci jedan drugi, uz male razlike:

- Izuzeci se obicno odnose na sinhrone dogadjaje, generisane od strane samog procesa (npr ilegalna instrukcija, deljenje sa 0, ilegalni pristup memoriji, sistemski poziv)

Uzroci izuzetaka

- Pojmovi izuzetak i prekid se uglavnom koriste zamenjujuci jedan drugi, uz male razlike:

- Izuzeci se obicno odnose na sinhrone dogadjaje, generisane od strane samog procesa (npr ilegalna instrukcija, deljenje sa 0, ilegalni pristup memoriji, sistemski poziv)
- Prekidi se obicno odnose na asinhrone dogadjaje, generisane od strane U/I uredjaja (tajmer istekao, pritisak tastera, primljen paket, ..)

Uzroci izuzetaka

- Pojmovi izuzetak i prekid se uglavnom koriste zamenjujuci jedan drugi, uz male razlike:

- Izuzeci se obicno odnose na sinhrone dogadjaje, generisane od strane samog procesa (npr ilegalna instrukcija, deljenje sa 0, ilegalni pristup memoriji, sistemski poziv)
- Prekidi se obicno odnose na asinhrone dogadjaje, generisane od strane U/I uredjaja (tajmer istekao, pritisak tastera, primljen paket, ..)

- Koristimo termin izuzetak za oba tipa dogadjaja, a specificiramo sinhroni izuzetak za sinhrone dogadjaje

Obrada izuzetaka

- Kada se izuzetak desi, procesor:

Obrada izuzetaka

- Kada se izuzetak desi, procesor:
  - Zaustavlja trenutni proces na instrukciji I_i, nakon sto zavrsi sve instrukcije do I_{i-1}

Obrada izuzetaka

- Kada se izuzetak desi, procesor:
  - Zaustavlja trenutni proces na instrukciji I_i, nakon sto zavrsi sve instrukcije do I_{i-1} *(precise exceptions)*
  - Cuva PC instrukcije I_i kao i uzrok nastanka izuzetka u specijalni (privilegovanim) registrima

Obrada izuzetaka

- Kada se izuzetak desi, procesor:
  - Zaustavlja trenutni proces na instrukciji I_i, nakon sto zavrsi sve instrukcije do I_{i-1} *(precise exceptions)*
  - Cuva PC instrukcije I_i kao i uzrok nastanka izuzetka u specijalni (privilegovanim) registrima
  - Prelazi u nadzirani mod, zabranjuje prekide, i prepusta kontrolu prethodno specifiranoj rutini za obradu izuzetka PC

Obrada izuzetaka

- Kada se izuzetak desi, procesor:
  - Zaustavlja trenutni proces na instrukciji I_i, nakon sto zavrsi sve instrukcije do I_{i-1} *(precise exceptions)*
  - Cuva PC instrukcije I_i kao i uzrok nastanka izuzetka u specijalni (privilegovanim) registrima
  - Prelazi u nadzirani mod, zabranjuje prekide, i prepusta kontrolu prethodno specifiranoj rutini za obradu izuzetka PC
- Nakon sto OS kernel zavrsi obradu izuzetka, on vraca kontrolu procesu koji nastavlja instrukciju I_i
  - Izuzetak je transparentan sa stanovista procesa!

Obrada izuzetaka

- Kada se izuzetak desi, procesor:
  - Zaustavlja trenutni proces na instrukciji I_i, nakon sto zavrsi sve instrukcije do I_{i-1} *(precise exceptions)*
  - Cuva PC instrukcije I_i kao i uzrok nastanka izuzetka u specijalni (privilegovanim) registrima
  - Prelazi u nadzirani mod, zabranjuje prekide, i prepusta kontrolu prethodno specifiranoj rutini za obradu izuzetka PC
- Nakon sto OS kernel zavrsi obradu izuzetka, on vraca kontrolu procesu koji nastavlja instrukciju I_i
  - Izuzetak je transparentan sa stanovista procesa!
- Ako je izuzetak posledica ilegalne operacije od strane programa koji ne moze biti korigovan (npr ilegalni pristup memoriji), OS terminira proces

Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

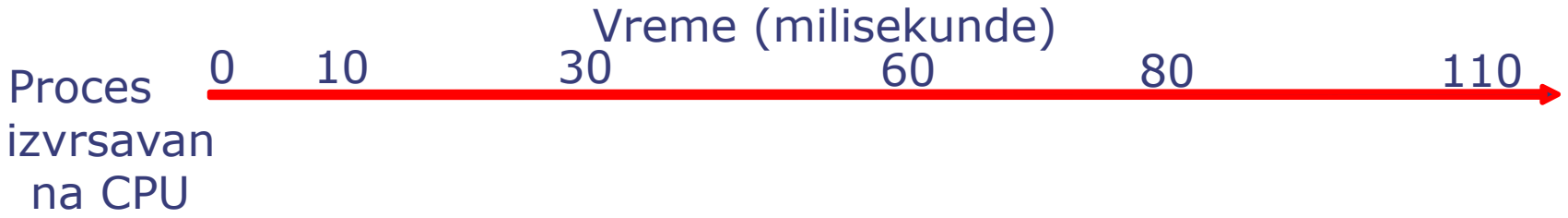
- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga

Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

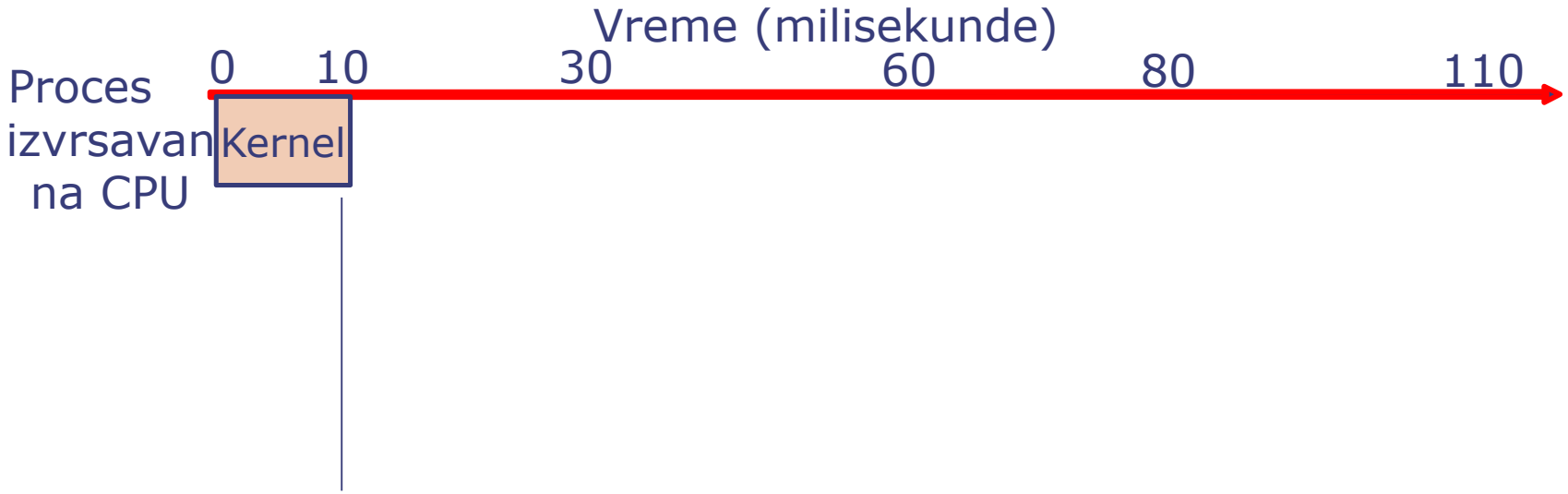


Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

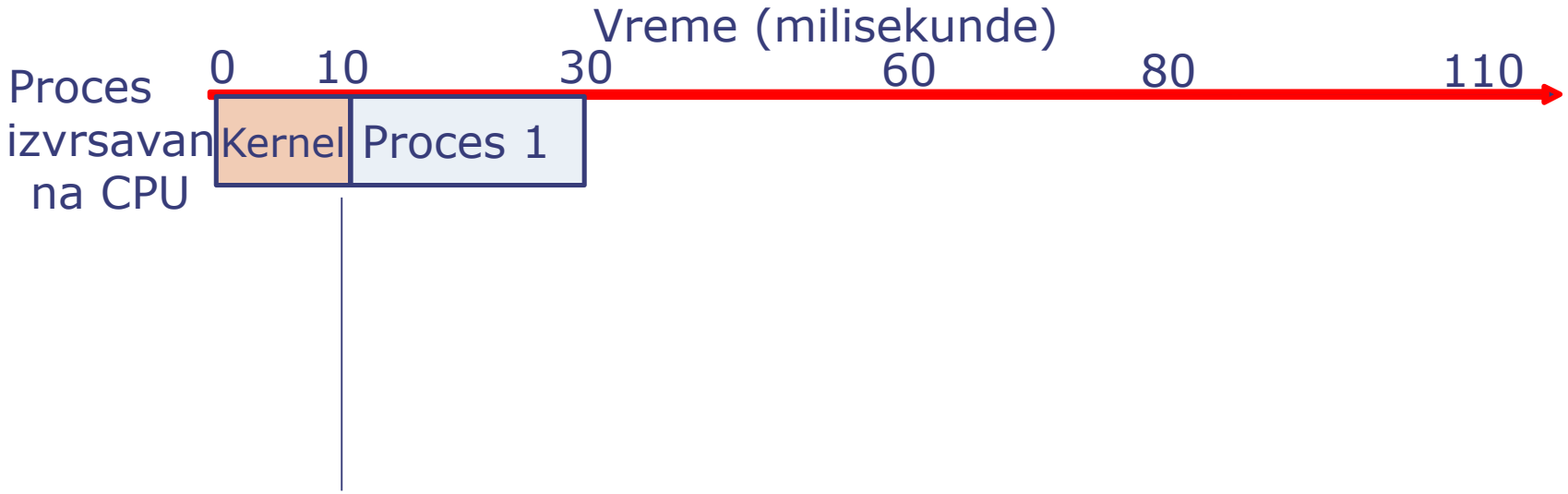
Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



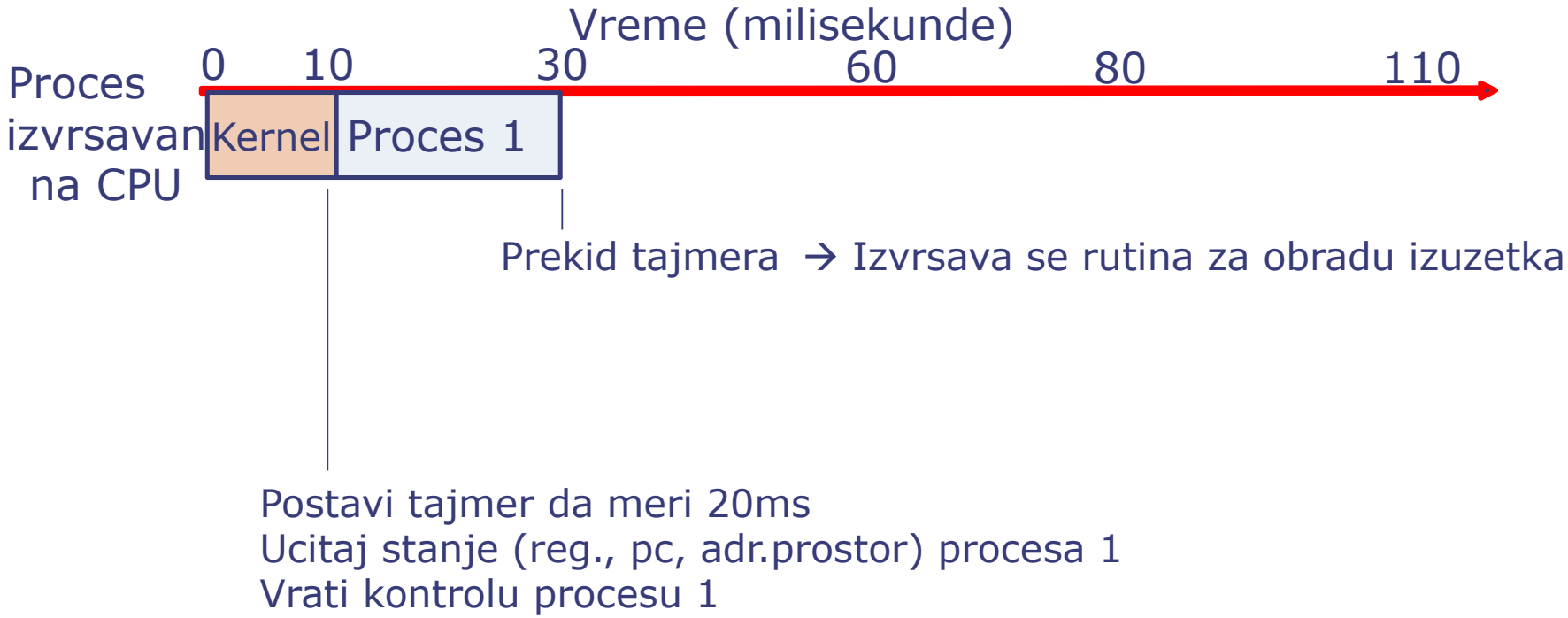
Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



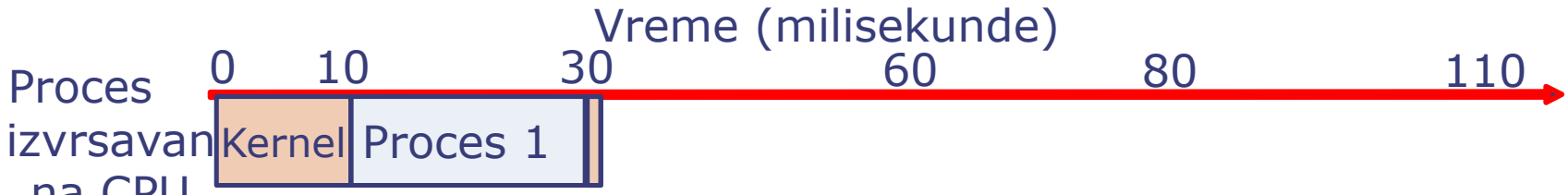
Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

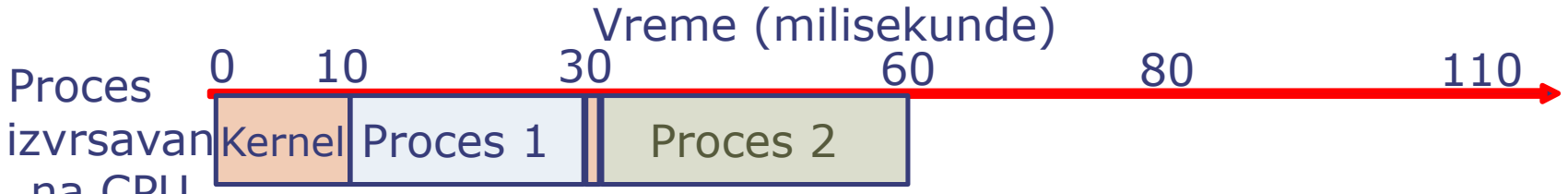


Prekid tajmera → Izvrsava se rutina za obradu izuzetka
Sacuvaj stanje procesa 1
Odluci da aktiviras proces 2
Postavi tajmer da meri 30ms
Ucitaj stanje procesa 2, vrati kontrolu njemu

Postavi tajmer da meri 20ms
Ucitaj stanje (reg., pc, adr.prostor) procesa 1
Vrati kontrolu procesu 1

Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

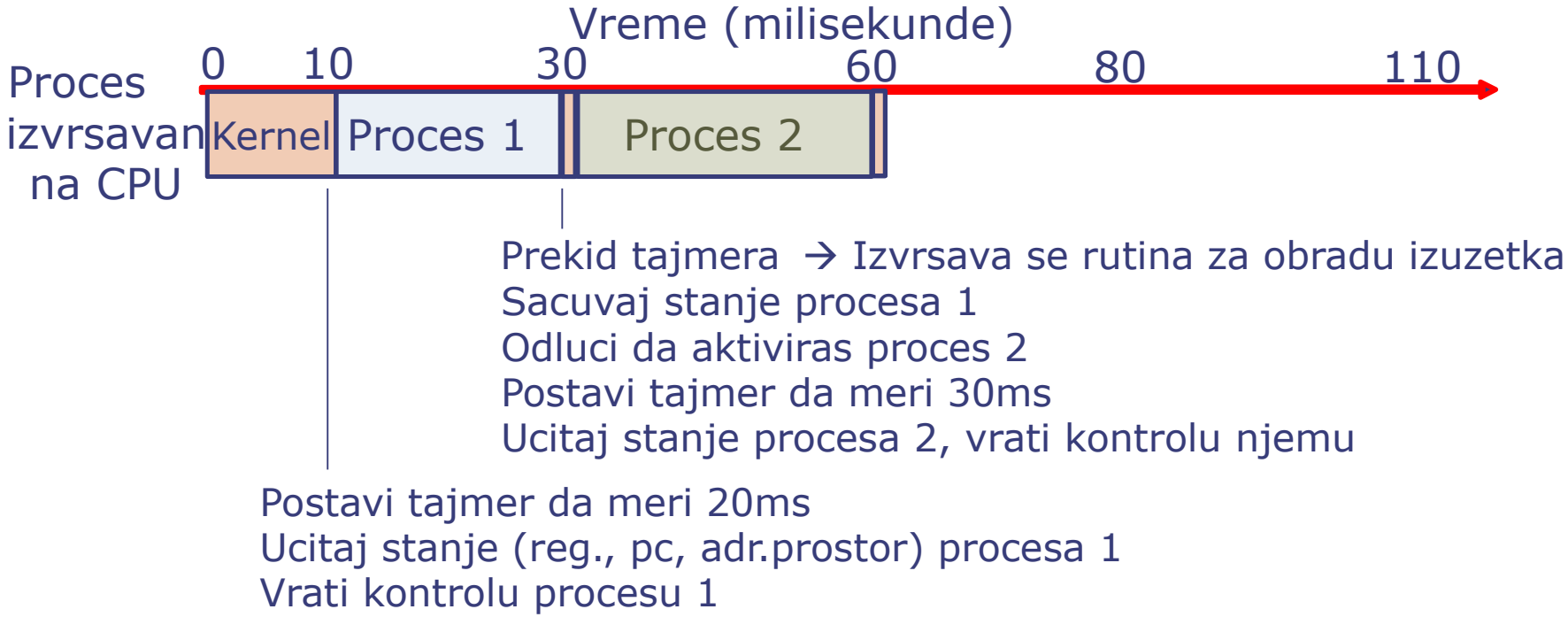


Prekid tajmera → Izvrsava se rutina za obradu izuzetka
Sacuvaj stanje procesa 1
Odluci da aktiviras proces 2
Postavi tajmer da meri 30ms
Ucitaj stanje procesa 2, vrati kontrolu njemu

Postavi tajmer da meri 20ms
Ucitaj stanje (reg., pc, adr.prostor) procesa 1
Vrati kontrolu procesu 1

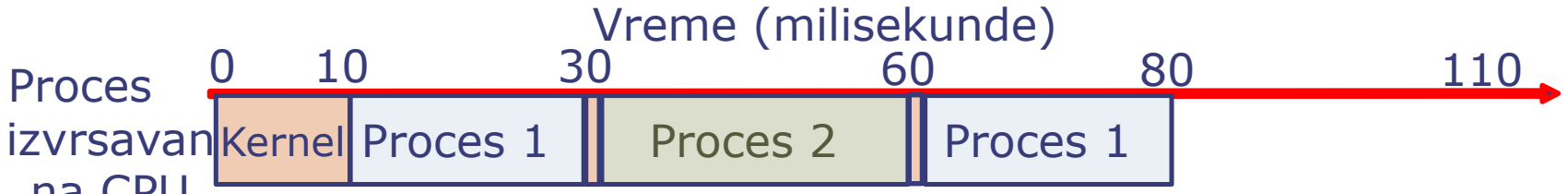
Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.

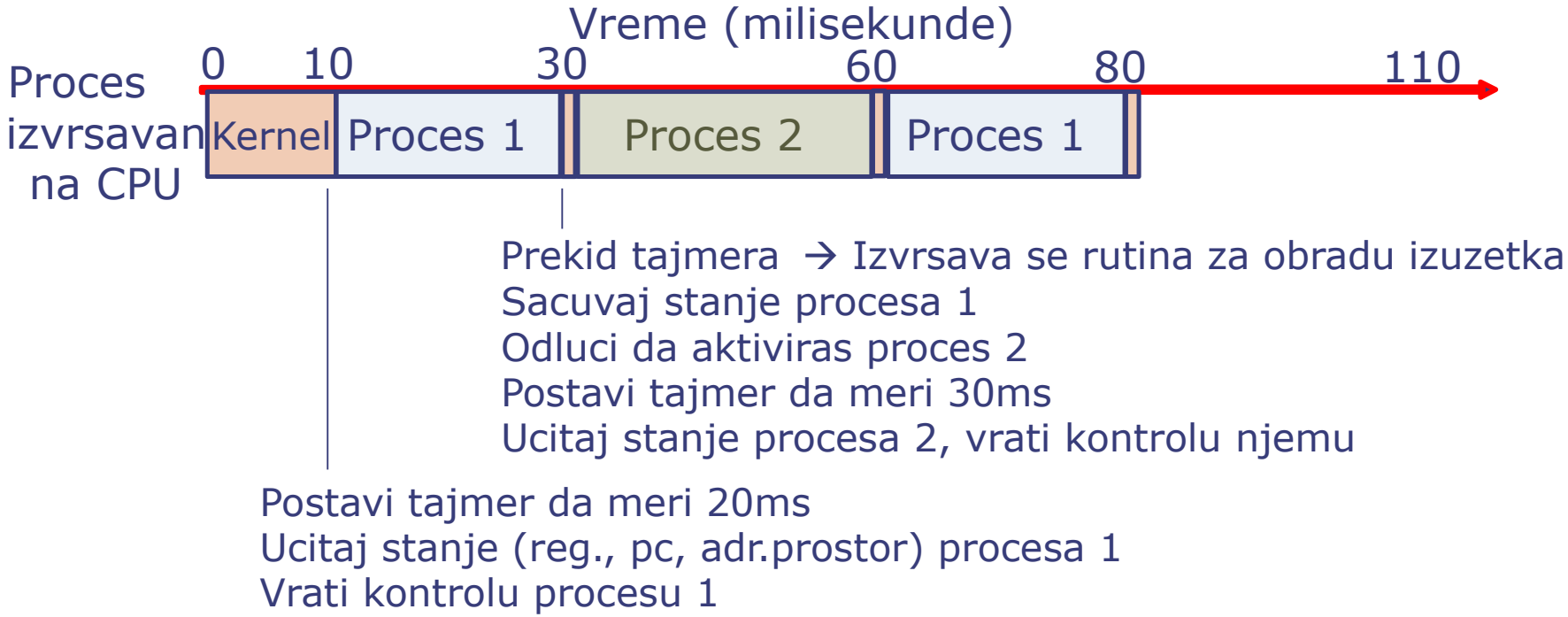


Prekid tajmera → Izvrsava se rutina za obradu izuzetka
Sacuvaj stanje procesa 1
Odluci da aktiviras proces 2
Postavi tajmer da meri 30ms
Ucitaj stanje procesa 2, vrati kontrolu njemu

Postavi tajmer da meri 20ms
Ucitaj stanje (reg., pc, adr.prostor) procesa 1
Vrati kontrolu procesu 1

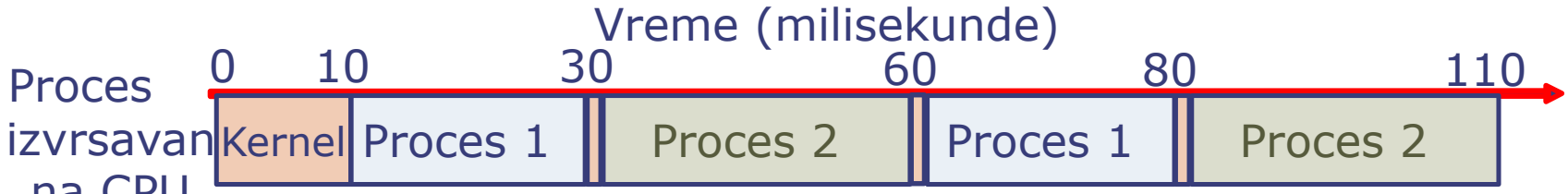
Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



Case Study 1: CPU rasporedjivanje
Omoguceno prekidima od strane tajmera

- OS kernel rasporedjuje procese na CPU
  - Svaki proces dobija period CPU vremena
  - Proces ne moze da koristi vise CPU vremena od toga
- Ovo je omoguceno zahvaljujuci prekidima tajmera
  - Kernel postavlja tajmer, koji generise prekid nekon odredjenog vrem.



Prekid tajmera → Izvrsava se rutina za obradu izuzetka
Sacuvaj stanje procesa 1
Odluci da aktiviras proces 2
Postavi tajmer da meri 30ms
Ucitaj stanje procesa 2, vrati kontrolu njemu

Postavi tajmer da meri 20ms
Ucitaj stanje (reg., pc, adr.prostor) procesa 1
Vrati kontrolu procesu 1

Case Study 2: Emulacija instrukcija

Omogucena izuzetkom usled ilegalne instrukcije

- mul x1, x2, x3 je instrukcija koja postoji u RISC-V ‘M’ ekstenziji (x1 := x2 * x3)
  - Ako ‘M’ nije implementirana, to je ilegalna instrukcija

Case Study 2: Emulacija instrukcija

Omogucena izuzetkom usled ilegalne instrukcije

- mul x1, x2, x3 je instrukcija koja postoji u RISC-V ‘M’ ekstenziji (x1 := x2 * x3)
  - Ako ‘M’ nije implementirana, to je ilegalna instrukcija
- Sta se desava ukoliko se izvrsava kod sa RISC-V ‘M’ masine na RISC-V masini?

Case Study 2: Emulacija instrukcija

Omogucena izuzetkom usled ilegalne instrukcije

- mul x1, x2, x3 je instrukcija koja postoji u RISC-V ‘M’ ekstenziji (x1 := x2 * x3)
  - Ako ‘M’ nije implementirana, to je ilegalna instrukcija
- Sta se desava ukoliko se izvrsava kod sa RISC-V ‘M’ masine na RISC-V masini?
  - mul izaziva izuzetak usled ilegalne instrukcije

Case Study 2: Emulacija instrukcija

Omogucena izuzetkom usled ilegalne instrukcije

- mul x1, x2, x3 je instrukcija koja postoji u RISC-V ‘M’ ekstenziji (x1 := x2 * x3)
  - Ako ‘M’ nije implementirana, to je ilegalna instrukcija
- Sta se desava ukoliko se izvrsava kod sa RISC-V ‘M’ masine na RISC-V masini?
  - mul izaziva izuzetak usled ilegalne instrukcije
- Rutina za obradu izuzetka preuzima kontrolu i terminira proces… ili mozda emulira instrukciju!

Case Study 2: Emulacija instrukcija

Omogucena izuzetkom usled ilegalne instrukcije

- mul x1, x2, x3 je instrukcija koja postoji u RISC-V ‘M’ ekstenziji (x1 := x2 * x3)
  - Ako ‘M’ nije implementirana, to je ilegalna instrukcija
- Sta se desava ukoliko se izvrsava kod sa RISC-V ‘M’ masine na RISC-V masini?
  - mul izaziva izuzetak usled ilegalne instrukcije
- Rutina za obradu izuzetka preuzima kontrolu i terminira proces… ili mozda emulira instrukciju!

Emuliranje nepodrzane instrukcije

Proces 1 kod:

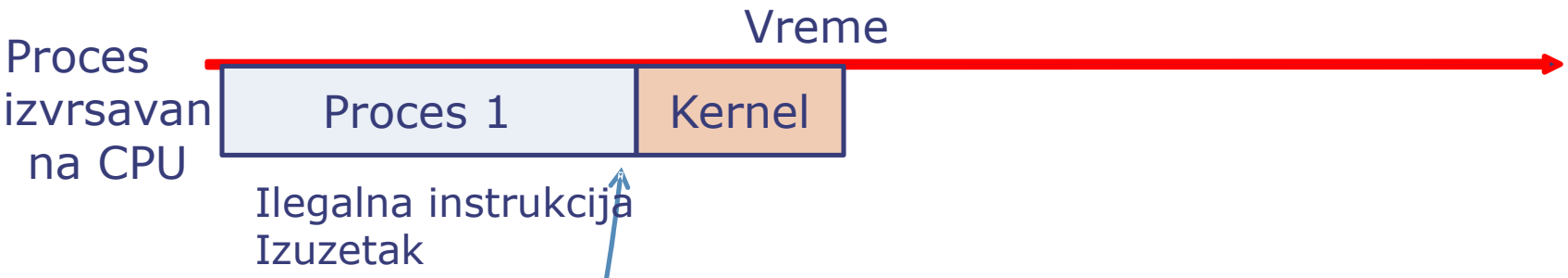
```
…
add a3, a2, a1
mul a4, a3, a2
xora5, a4, a3
…
```

Emuliranje nepodrzane instrukcije

Proces 1 kod:

```
…
add a3, a2, a1
mul a4, a3, a2
xora5, a4, a3
…
```

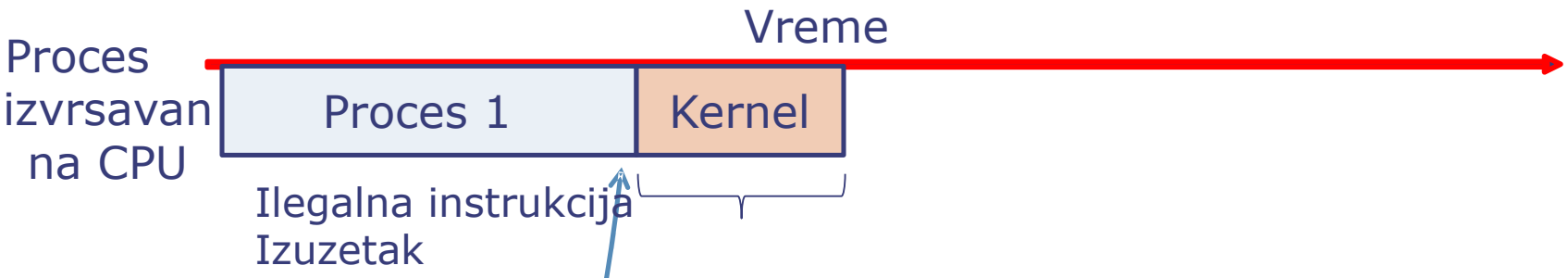
Emuliranje nepodrzane instrukcije



Proces 1 kod:

```
…
add a3, a2, a1
mul a4, a3, a2
xora5, a4, a3
…
```

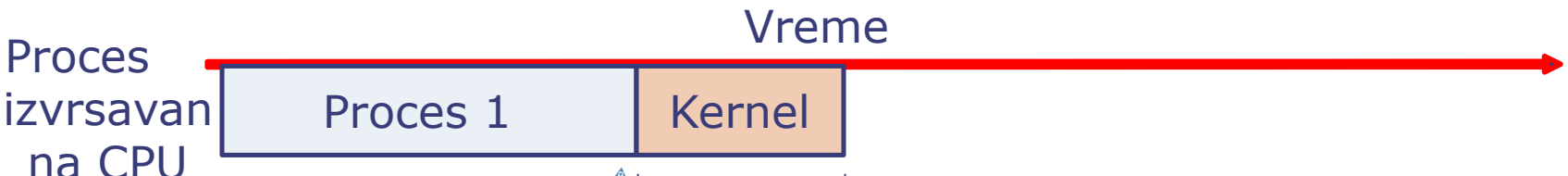
Emuliranje nepodrzane instrukcije



Proces 1 kod:

```
…
add a3, a2, a1
mul a4, a3, a2
xora5, a4, a3
…
```

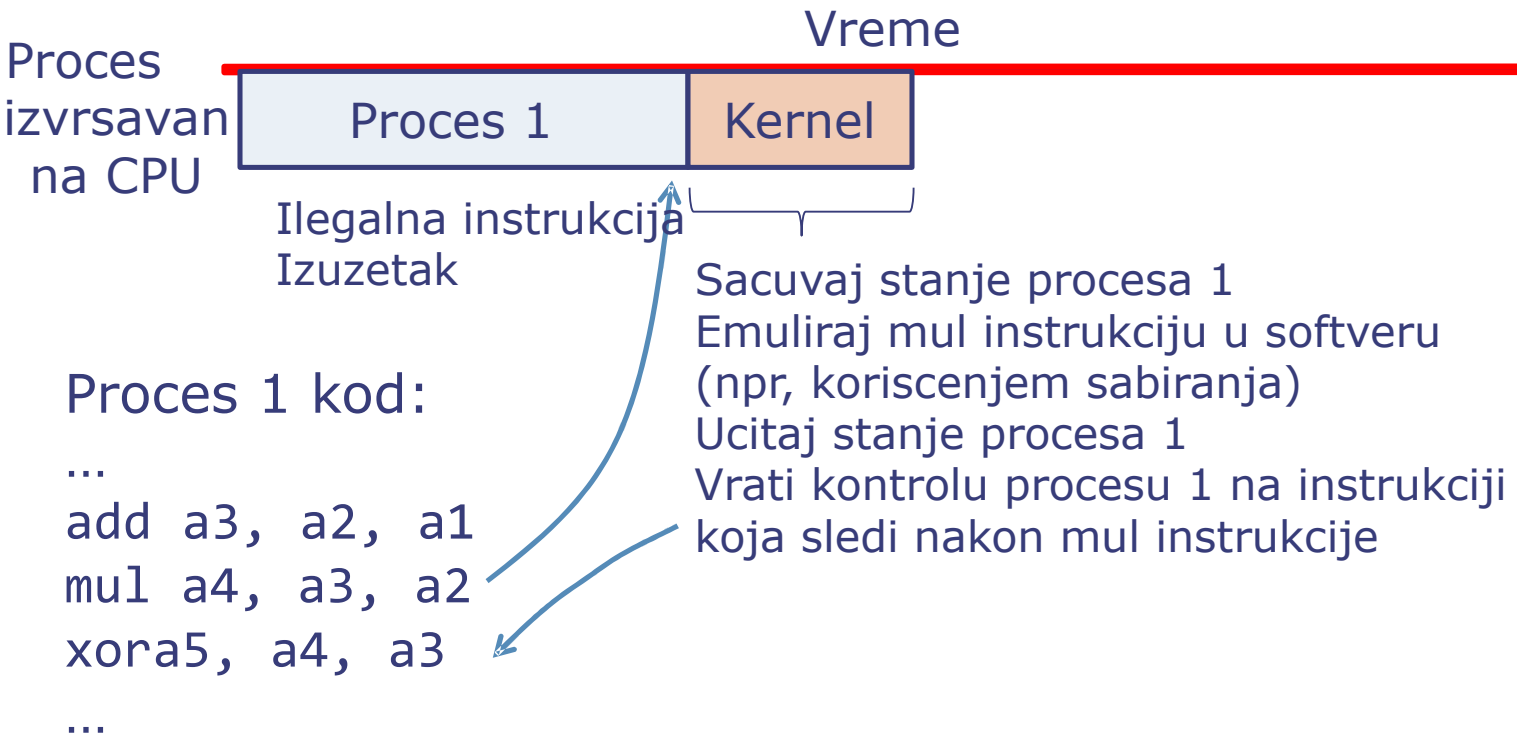
Emuliranje nepodrzane instrukcije



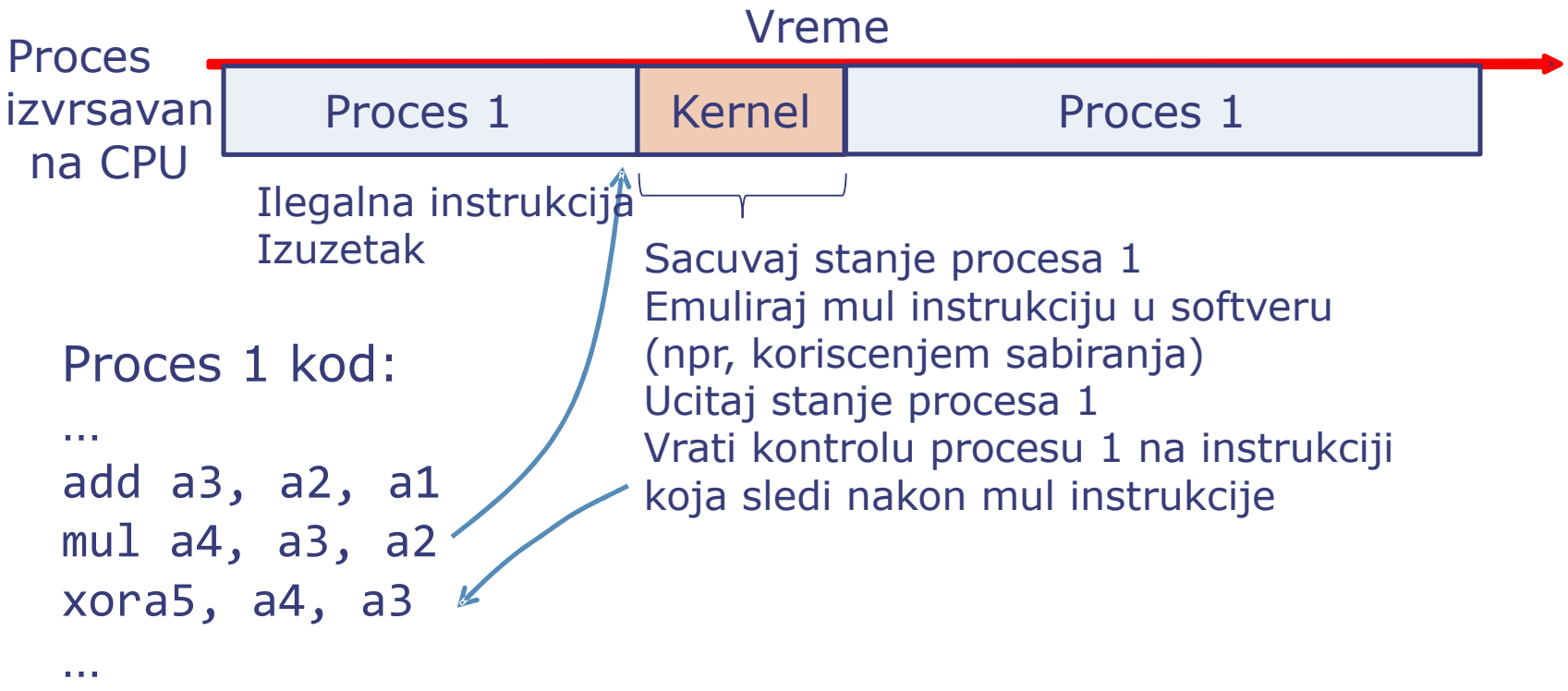
Proces 1 kod:

```
…
add a3, a2, a1
mul a4, a3, a2
xora5, a4, a3
…
```

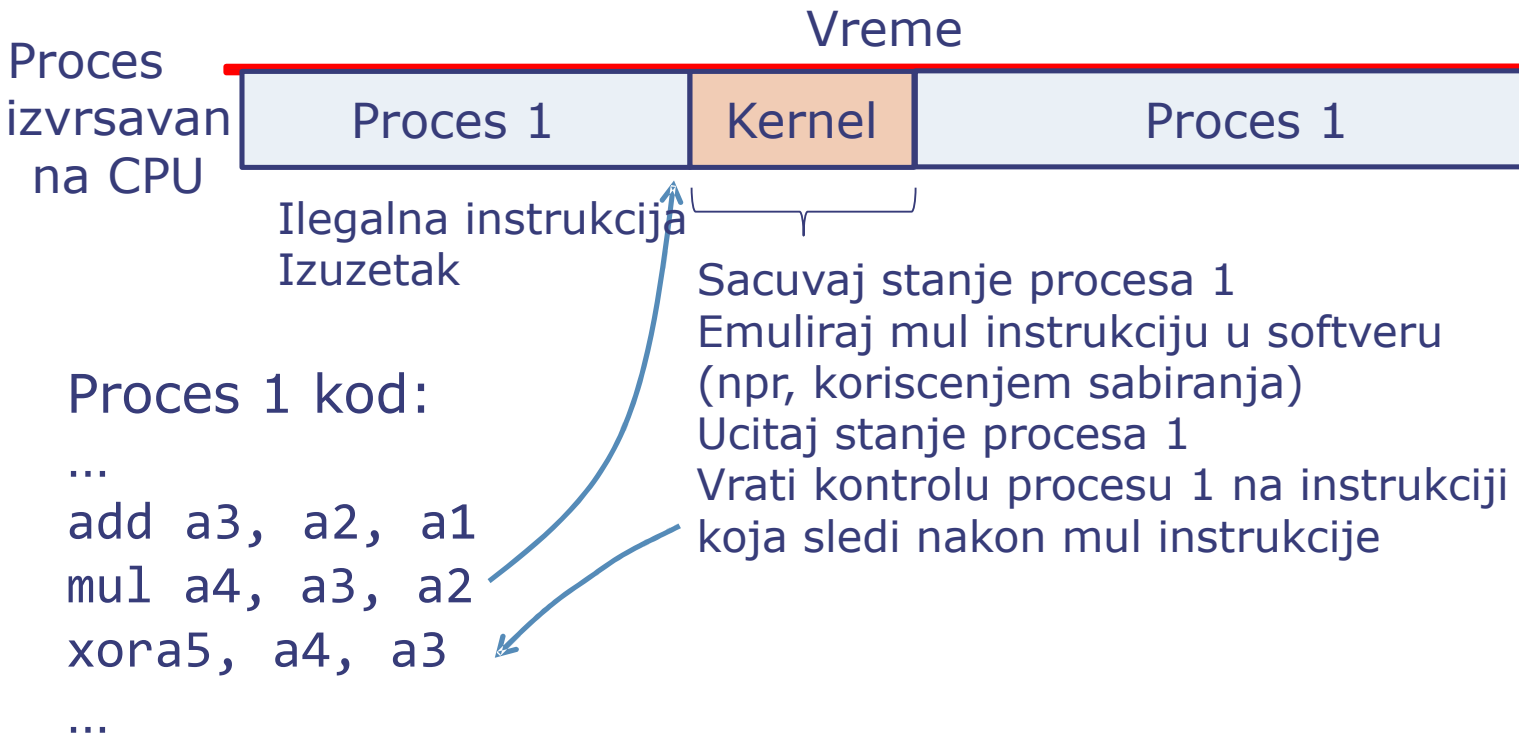
Emuliranje nepodrzane instrukcije



Emuliranje nepodrzane instrukcije

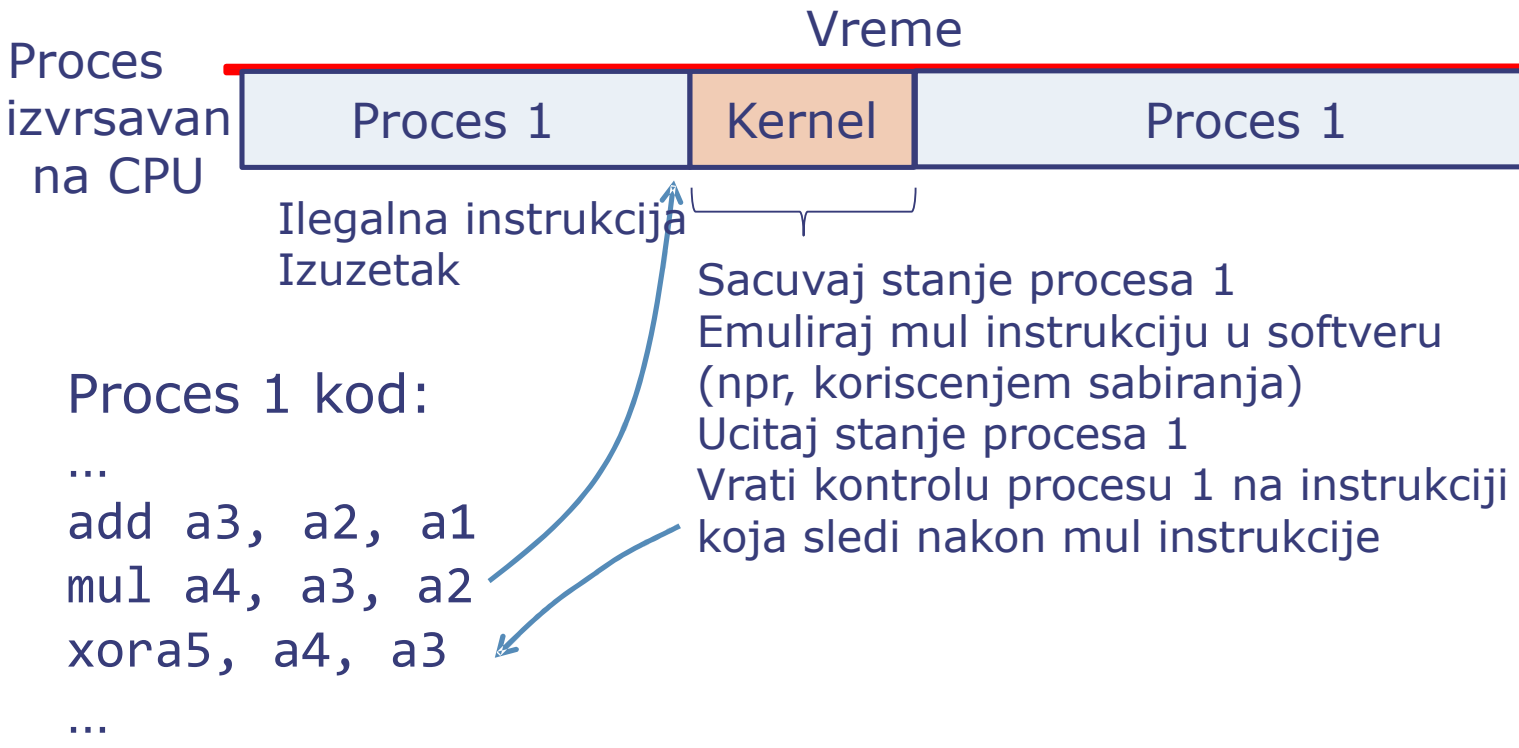


Emuliranje nepodrzane instrukcije



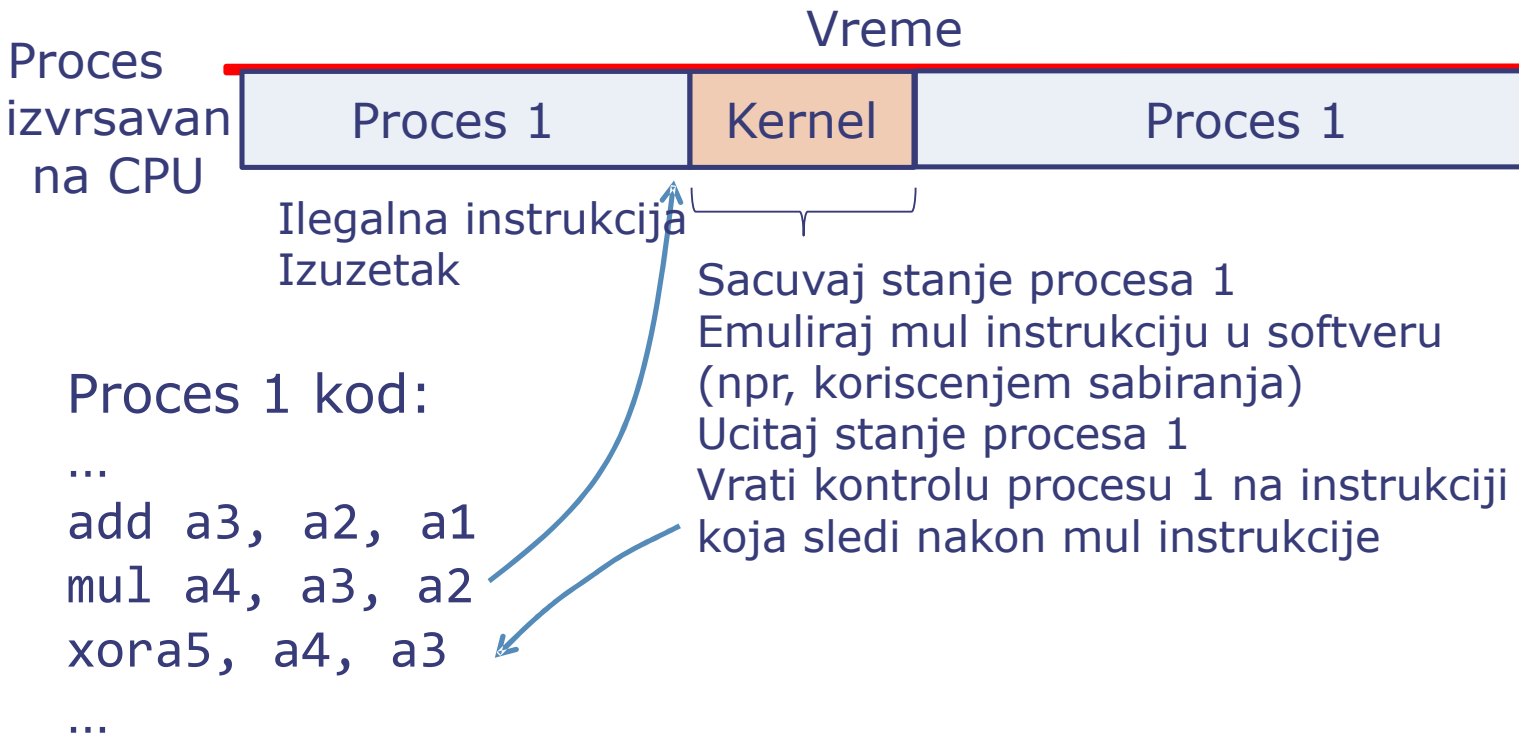
- Rezultat: Program veruje da se izvrsava na RISC-V ‘M’ procesoru, iako se zapravo izvrsava na procesoru bez ‘M’ ekstenzije

Emuliranje nepodrzane instrukcije



- Rezultat: Program veruje da se izvrsava na RISC-V ‘M’ procesoru, iako se zapravo izvrsava na procesoru bez ‘M’ ekstenzije
  - *Problem?*

Emuliranje nepodrzane instrukcije



- Rezultat: Program veruje da se izvrsava na RISC-V ‘M’ procesoru, iako se zapravo izvrsava na procesoru bez ‘M’ ekstenzije
  - *Problem?* *Mnogo sporije nego hardversko mnozenje*