Виртуелизација

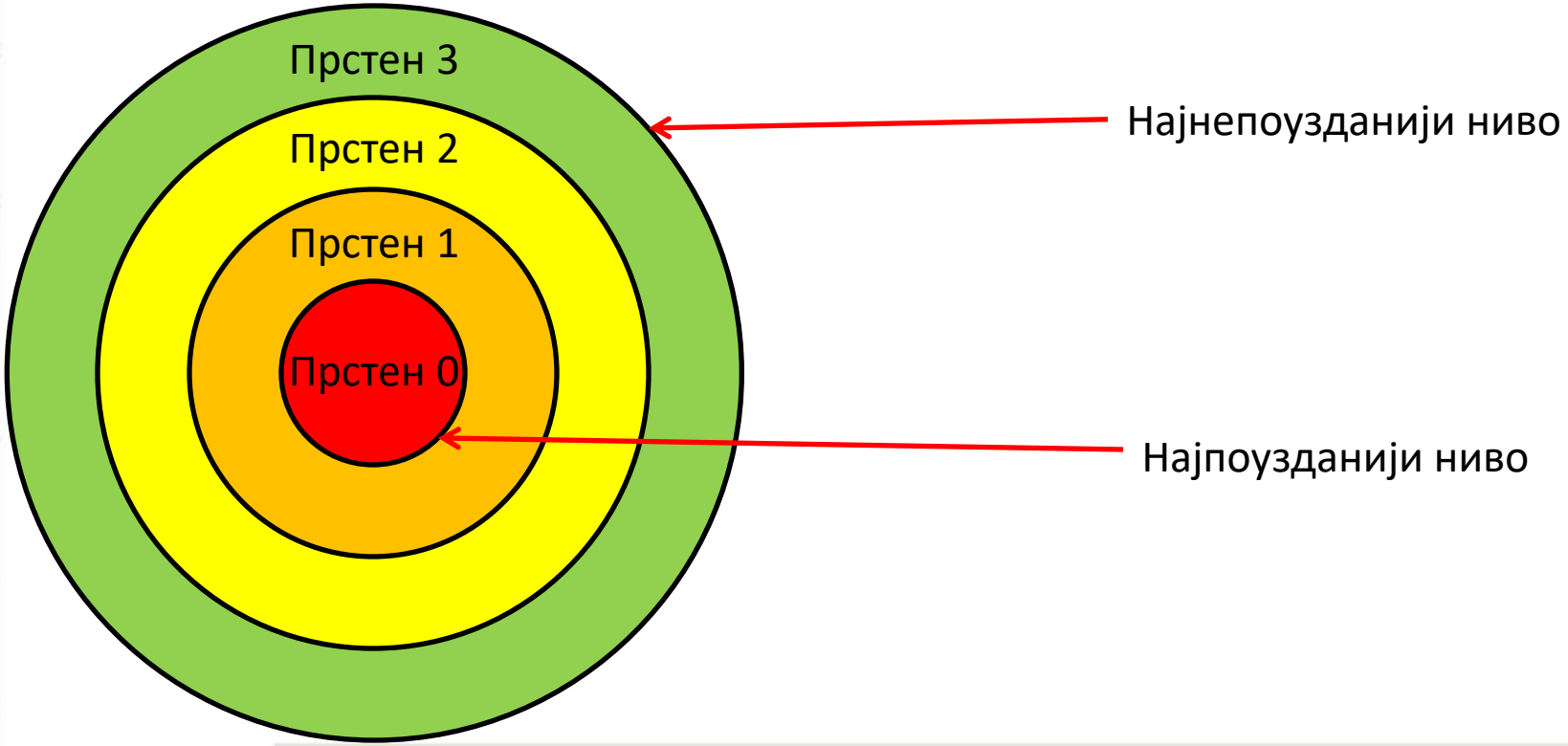
Садржај

- Нивои привилегија
- Виртуелизација
  - Виртуелизација процесора
  - Виртуелизација меморије
  - Виртуелизација уређаја

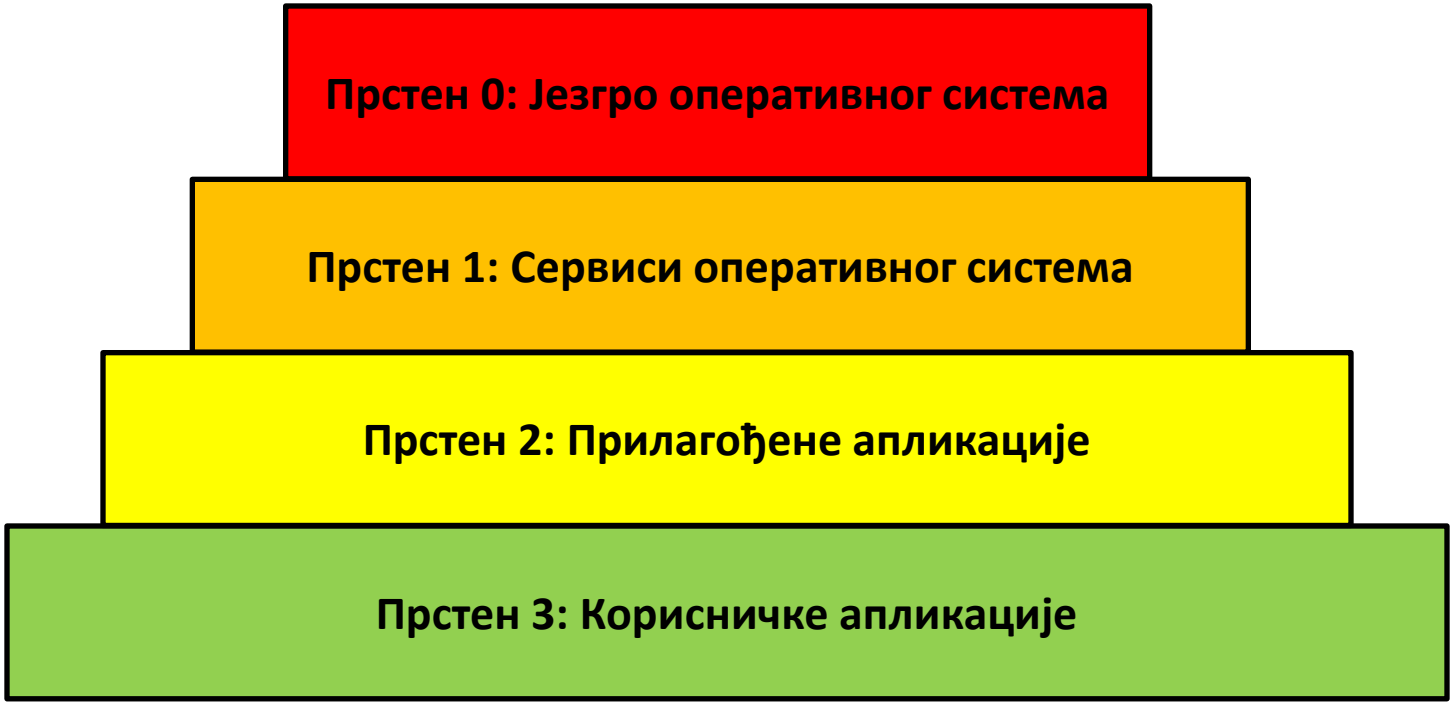
Нивои привилегија

- Заштићени режим рада процесора (*protected-mode*) омогућава да се наметне ограничење слободе да одређени софтвер предузме одређене радње
- Код x86 архитектуре подржана су четири различита нивоа привилегија
- Организација је заснована на „концентричним прстеновима” - *rings*
- Централни прстен има највеће привилегије, а привилегије се смањују како се прстенови удаљавају од центра

Четири нивоа привилегија – x86



Предложени начин коришћења – x86



Unix/Linux и Windows – x86

Прелазак између нивоа

- Прелазак из спољног прстена на унутрашњи прстен омогућен је употребом посебне контролне структуре („капија позива” ‘*call gate*’)
- Капија је дефинисана преко структуре података која се налази у системском меморијском сегменту који обично није доступан за модификације
- Прелаз из унутрашњег прстена на спољни прстен није ни приближно тако строго контролисан

Дељење података

- Позиви функција обично захтевају да две одвојене функције (рутине) деле неке податке (нпр. вредности параметара прелазе из позивне рутине у позвану рутину)
- Да би подржао повратак и рекурзију, сегмент стека процесора се често користи као подручје за складиштење са заједничким приступом
- Али међу рутинама са различитим нивоима привилегија ово би могло створити сигурносну рупу

Пример

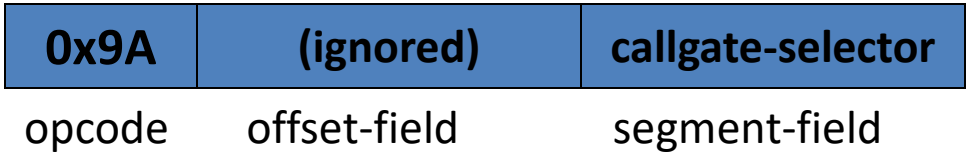
- Претпоставимо да процедура која се извршава у прстену 3 позива процедуру која се извршава у прстену 2
- Процедура прстен 2 користи део свог простора на стеку да би креирала аутоматске променљиве које користи за привремени радни простор
- По повратку процедура у прстену 3 би могла да испита све вредности које су остале у овом радном простору прстена 2

Изолација података

- Да би се заштитили од ненамерног дељења привилегованих информација, на сваком нивоу привилегија су обезбеђени различити стекови
- Сходно томе, сваки прелазак из једног прстена на други мора нужно бити праћен обавезном операцијом пребацивања стека
- Процесор омогућава аутоматско пребацивање стекова и копирање вредности параметара

Позиви између различитих нивоа привилегија

- Када мање привилегована процедура жели да позове привилегованију рутину, то чини употребом машинске инструкције за далеки позив, *far call*
(познате и као дуги позив, *long call,* у ГНУ терминологији)



- Асемблерска инструкција:

```
lcall   $callgate-selector, $0
```

Како се ова инструкција извршава

- Када процесор крене да извршава дату инструкцију користи операнд дате инструкције, селектор, да потражи дескриптор у глобалној табели дескриптора (*Global Descriptor Table – GDT*) (или у локалној табели дескриптора (*Local Descriptor Table – LDT*)
- Уколико се ради о дескриптору који дозвољава прелазак, и ако је прелазак дозвољен (ако је CPL ≤ DPL), тада процесор обавља скуп радњи који треба да омогуће промену нивоа привилегија
- Текући ниво привилегија (*Current Privilege Level – CPL*) је одређен са два бита најмање тежине регистра CS (такође и SS регистра)

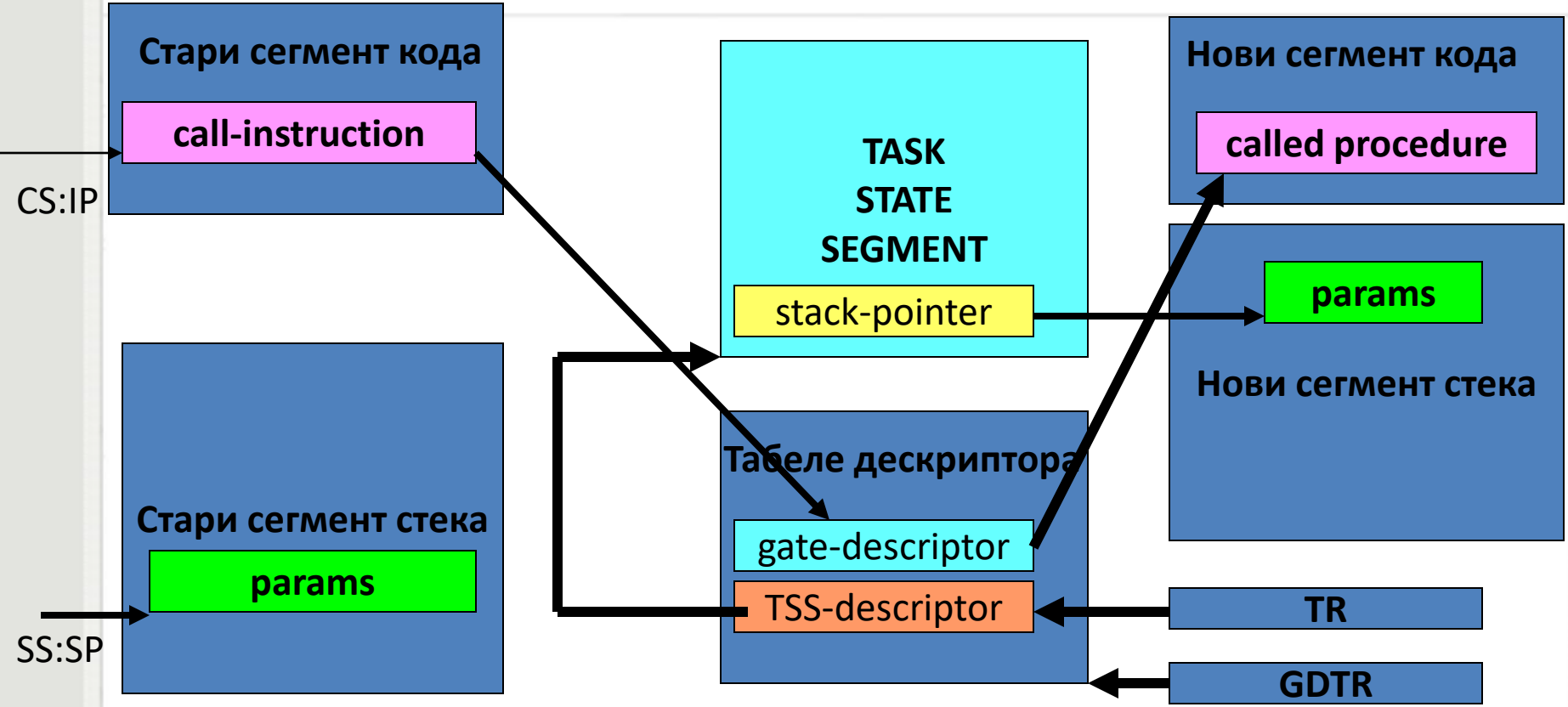
Скуп радљи које обавља процесор

- Поставља текућу вредност SS:SP регистра на нови стек сегмент
- Копира специфицирани број параметара са старог стека на нови стек
- Поставља ажуриране вредности CS:IP регистра на нови стек
- Учитава нове вредности у CS:IP регистар из дескриптора и нову вредност у SS:SP регистар

Шта недостаје?

- Одакле долази нова вредност за SS:SP регистар? (Ово се не чува у дескриптору!)
- Ова вредност долази из специјализованог системског сегмента *TSS* (*Task State Segment*)
- Процесор учитава *TSS* сегмент референцирајући наменски регистар *TR* (*Task Register*)

Релације


Стари сегмент кода
call-instruction
CS:IP
TASK
STATE
SEGMENT
stack-pointer
Нови сегмент кода
called procedure
params
Нови сегмент стека
Стари сегмент стека
params
SS:SP
Табеле дескриптора
gate-descriptor
TSS-descriptor
TR
GDTR

Повратак у спољашњи прстен

- Користи се инструкција за далеки повратак: lret
  - Рестаурира CS:IP са текућег стека
  - Рестаурира SS:SP са текућег стека
- Или ту инструкцију са аргументом: lret $n
  - Рестаурира CS:IP са текућег стека
  - Уклања n бајтова параметара са стека
  - Рестаурира SS:SP са текућег стека

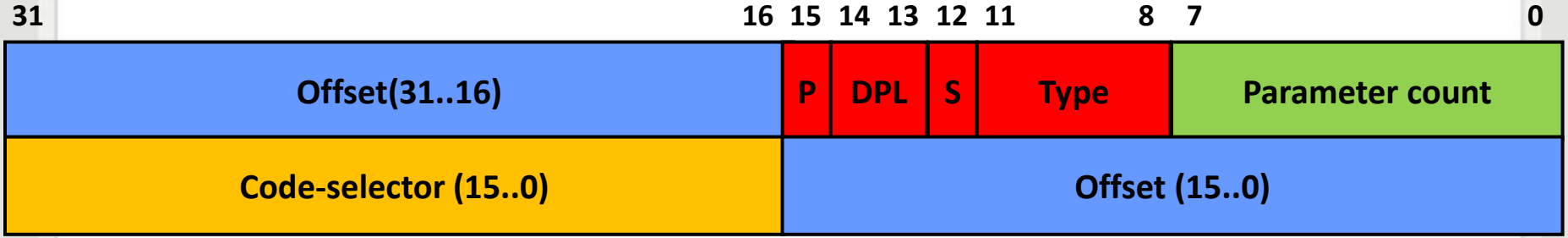
Инструкције за рад са привилегијама

Instruction	Description
LGDT	Loads an address of a GDT into GDTR
LLDT	Loads an address of a LDT into LDTR
LTR	Loads a Task Register into TR
MOV Control Register	Copy data and store in Control Registers
LMSW	Load a new Machine Status WORD
CLTS	Clear Task Switch Flag in Control Register CR0
MOV Debug Register	Copy data and store in debug registers
INVD	Invalidate Cache without writeback
INVLPG	Invalidate TLB Entry
WBINVD	Invalidate Cache with writeback
HLT	Halt Processor
RDMSR	Read Model Specific Registers (MSR)
WRMSR	Write Model Specific Registers (MSR)
RDPMC	Read Performance Monitoring Counter
RDTSC	Read time Stamp Counter

Глобална табела дескриптора

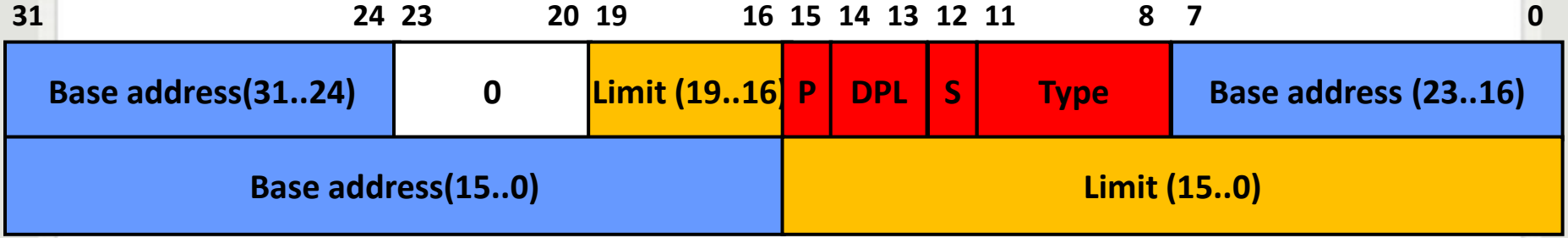
- Ова табела садржи информације о глобалним сегментима
- Не мора садржати само дескрипторе сегмената.
- Један улаз у табели је 8 бајтова
- Улаз може представљати
  - Меморијски сегмент
  - *Task State Segment (TSS)*
  - *Call Gate*
  - *Local Descriptor Table (LDT)*. Више ових табела се може наћи у GDT, али се само једна може користити у неком тренутку.

Капија позива – *call gate*



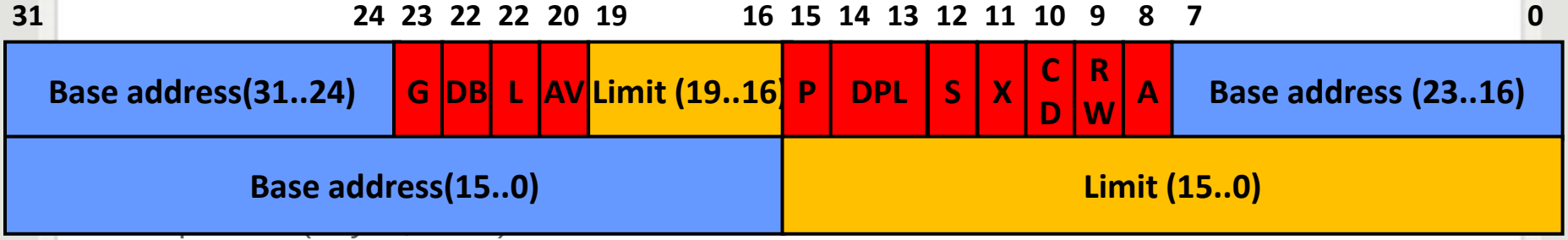
- P=present (1=yes, 0=no)
- DPL=Descriptor Privilege Level (0, 1, 2, 3)
- S=System (0)
- Type (‘0x4’ 16-bit call-gate, ‘0xC’ 32-bit call-gate)
- Parameter count (specifies how many parameter-values will be copied)
- Code-selector (specifies memory-segment containing procedure code)
- Offset (specifies the procedure’s entry-point within its code-segment)

System Segment-Descriptors



- P=present (1=yes, 0=no)
- DPL=Descriptor Privilege Level (0, 1, 2, 3)
- S=System (0)
- Type (0=reserved, 1=16-bit TSS (available), 2=LDT, 3=16-bit TSS (busy), 8=reserved, 9=32-bit TSS (available), A=reserved, B=32-bit TSS (busy))

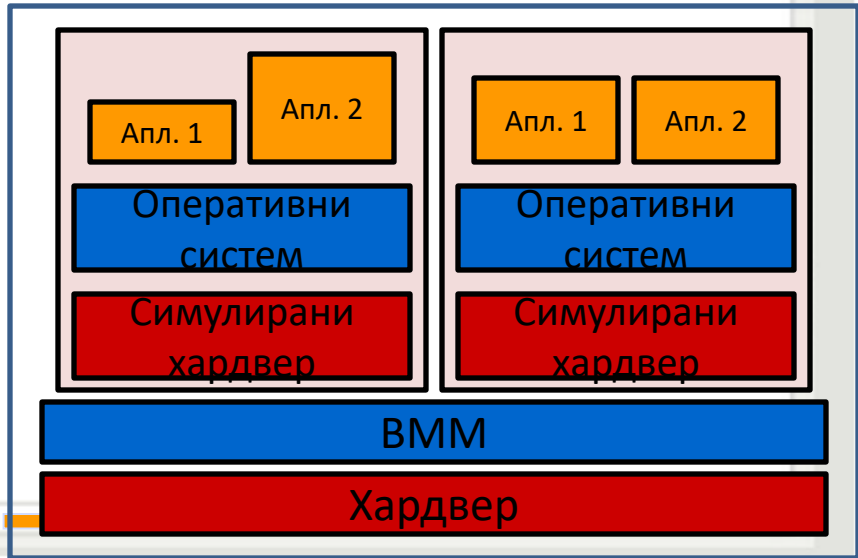
Segment-Descriptors



- DPL=Descriptor Privilege Level (0, 1, 2, 3)
- S=System (0=yes, 1=no)
- X=eXecutable (0=no, 1=yes)
- *code-segments:* R=Readable (0=no, 1=yes) C=Conforming (0=no, 1=yes)
- *data-segments:* W=Writable (0=no, 1=yes) D=expands-Down (0=no, 1=yes)
- A=Accessed (0=no, 1=yes)
- G=Granularity (0=byte, 1=4KB-page)
- DB=Default size (0=16-bit, 1=32-bit)
- L=Long-mode (i.e., 64-bit addressing) (0=no, 1=yes)
- AV=Available for user's purposes

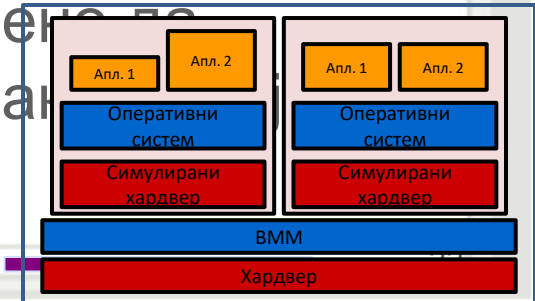
Виртуелизација

- Виртуелизација треба да да платформу која емулира хардверску платформу и да дозволи да више инстанци оперативног система користи ту платформу, као да имају пуни и ексклузивни приступ основном хардверу



Преглед

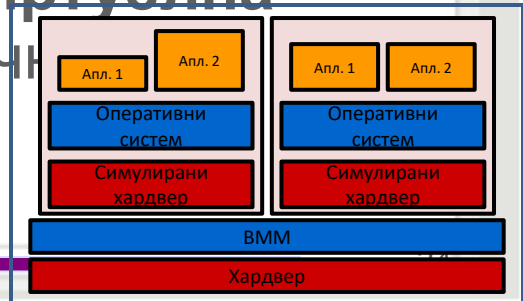
- Основна идеја:
  - апстрактни хардвер једног рачунара у неколико различитих окружења за извршење
  - Слично слојевитом приступу
  - Али слој ствара виртуелни систем (виртуелна машина или ВМ) на којем могу да се покрећу оперативни системи или апликације
- Једна физичка машина може истовремено да покреће више оперативних система, свак виртуелној машини



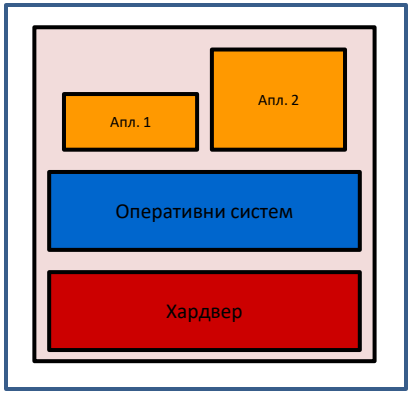
Виртуелизација

Најзначајније компоненте

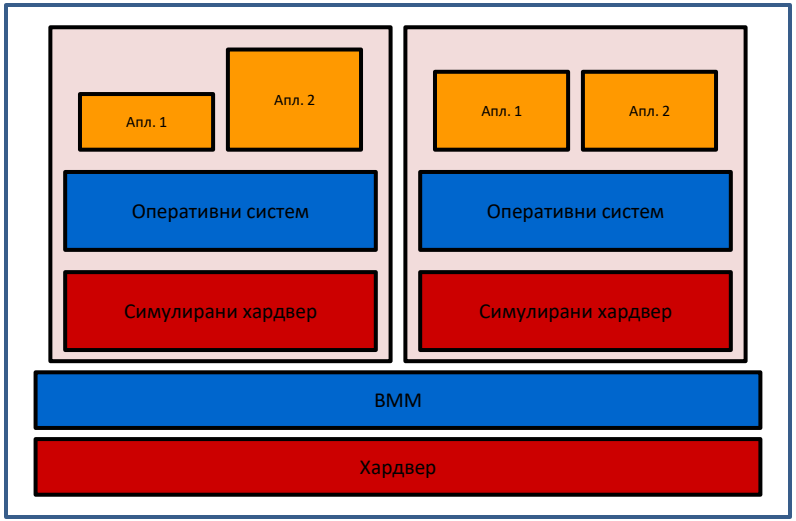
- **Домаћин** (***Host***) – хардверски систем на коме се све извршава
- ***Virtual machine manager*** (***VMM***) или ***hypervisor*** – креира и покреће виртуелне машине тако што пружа интерфејс који је **идентичан** као онај који пружа *host* (Изузета је техника паравиртуелизације)
- **Гост** (***Guest***) – процес коме се пружа **виртуелна** копија хардверског система. Ово је обичн оперативни систем.



Модели система



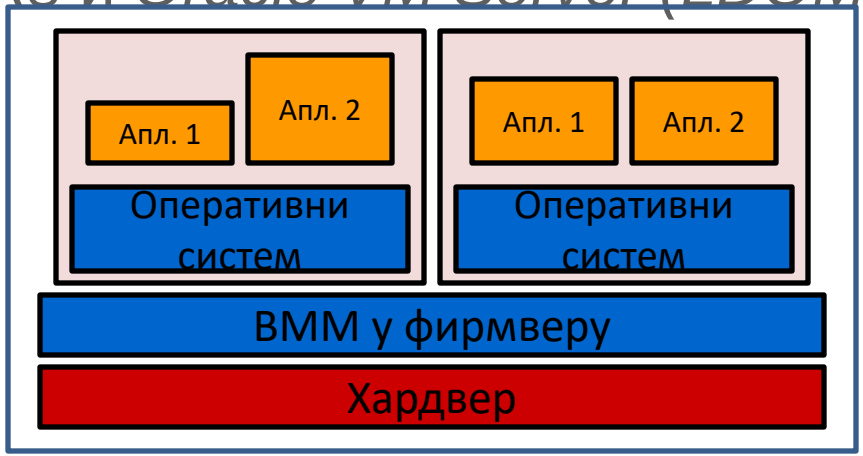
Нема виртуелизације



Има виртуелизације

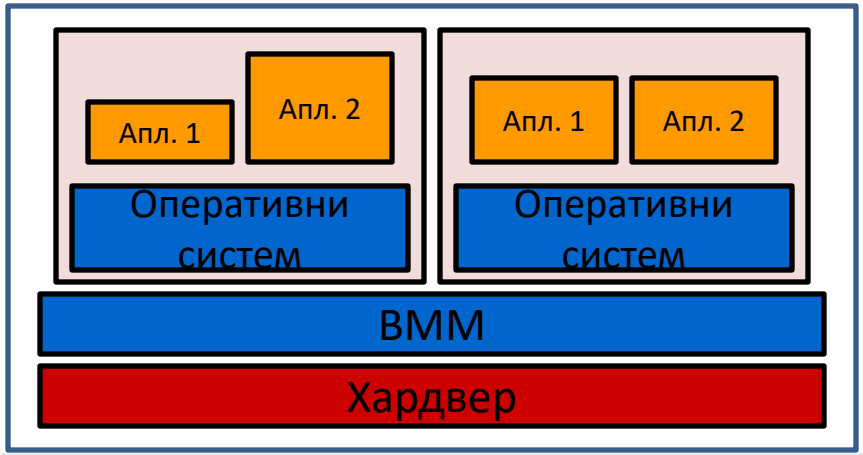
Имплементације

- **Тип 0** – Решење засновано на коришћењу минималних ресурса (хардверских и софтверских) тако да пружи могућност за креирање и одржавање виртуелних машина кроз фирмвер (*IBM LPARs и Oracle VM Server (LDOMs)*, ...)



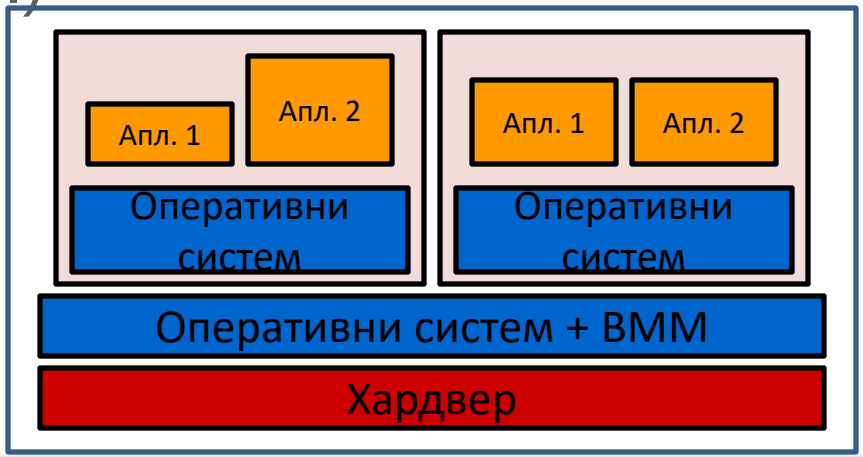
Имплементације

- **Тип 1 микрокернел** – Софтверска решења налик на оперативне системе који пружају виртуелизацију (*VMware ESX*, *Joyent SmartOS*, и *Citrix XenServer*, ...)



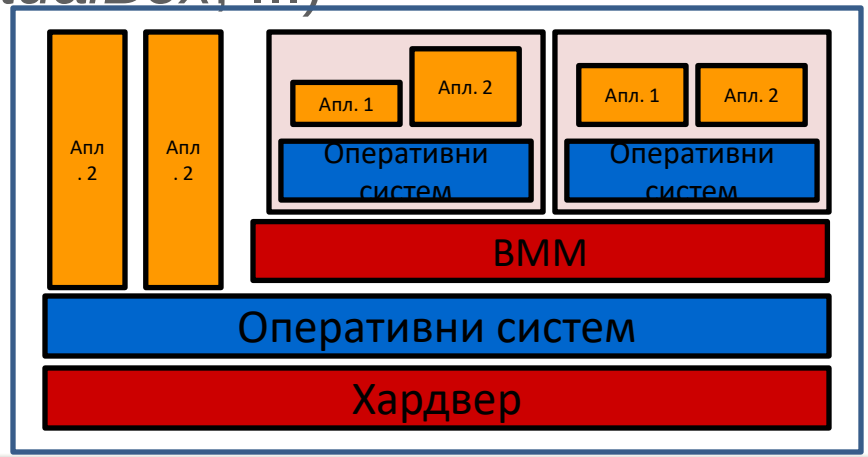
Имплементације

- **Тип 1 монолитни** – Представљају интеграцију општенаменског оперативног система и система за виртуелизацију
(*Microsoft Windows Server* са *HyperV* и *RedHat Linux* са *KVM*, ...)



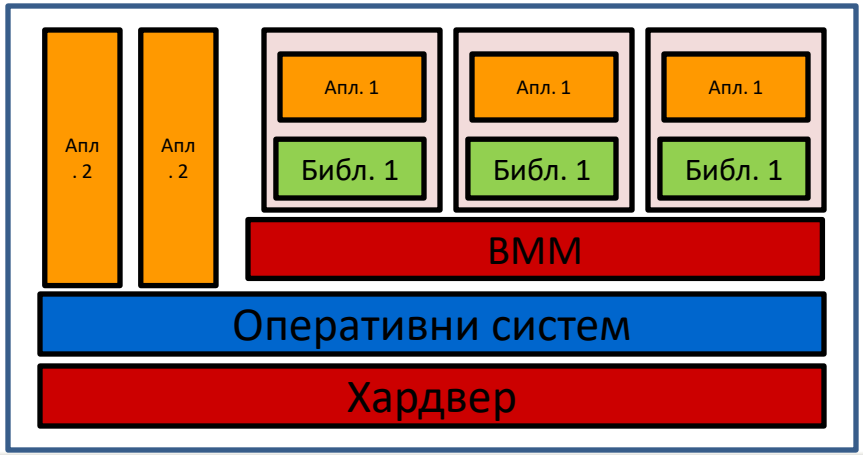
Имплементације

- **Тип 2** – Апликације које се извршавају на стандардним оперативним системима и пружају подршку за виртуелизацију гостујућим системима (*VMware Workstation* и *Fusion*, *Parallels Desktop*, и *Oracle VirtualBox*, ...)



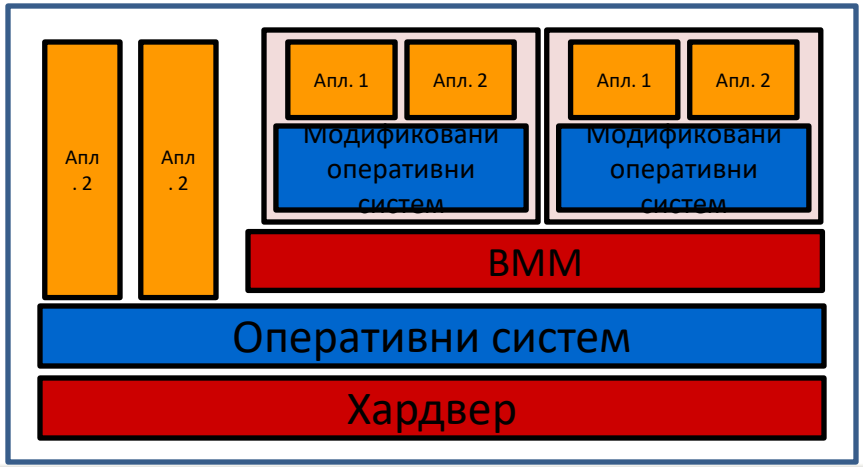
Имплементације

- **Контејнери** – Могућност да се истом оперативном систему покрене више изолованих корисничких простора
(*Docker*, *Kubernetis*, *LXC*, ...)



Имплементације

- **Паравиртуелизација** – Техника ког које је потребно да се гостујући оперативни систем модификује како би могао да сарађује са оперативним системом домаћина



Имплементације

- **Виртуелизација програмског окружења** – Окружење не виртуелизује стваран хардвер већ уместо тога креира оптимизовани виртуелни систем (*Oracle Java* и *Microsoft.Net*)
- **Емулација** – Софтвер који дозвољава апликацијама писаним за једно хардверско окружење да се извршавају на потпуно другачијем окружењу да се извршавају (може бизи и другачији процесор)
- …

Историјски преглед

- Први пут се појавио у ИБМ меинфреим рачунарима почетком седамдесетих
- Дозвољавао је да више корисника пакетски деле ресурсе *batch system*
- Формално дефинисање виртуелизације омогућило је да се користи и на другим окружењима
  - ВММ пружа окружење за програме које је у основи идентично оригиналној машини
  - Програми који се изводе у том окружењу показују само мања смањења перформанси
  - ВММ има потпуну контролу над системским ресурсима
- Крајем деведесетих Интелови процесори довољно брзо да истраживачи покушају да виртуелизују на рачунарима опште намене
  - Ксен и ВМваре створили су технологије које се користе и данас
  - Виртуелизација се проширила на многе оперативне системе и процесоре

Користи и особине

- Систем домаћина заштићен је од гостију, а гости заштићени једни од других
  - Тј. Вирус има мање шансе за ширење
  - Дељење је омогућено путем заједничког система датотека, мрежне комуникације
- Замрзни, обустави, покрени
  - Тада можете да се преместите или копирате негде другде и наставите
  - Снимак датог стања, способан да се врати у то стање
    - Неки системи омогућавају вишеструке снимке по госту
  - Клонирање се обавља тако што се направи копија и покрене и оригинал и копија
  - Погодно за истраживање ОС-а, бољу ефикасност развоја система
  - Покретање више различитих оперативних система на једној машини
    - Консолидација, развој апликације,…

Користи и особине

- Шаблони - креирање ОС + апликацију за ВМ, и њено пружање купцима, ово омогућава креирање више инстанци исте комбинације
- Миграција уживо - преместите активни ВМ са једног домаћина на други!
  - Нема прекида корисничког приступа
- Све те функције узете заједно -> рачунарство у облаку
  - Користећи АПИ, програми наводе облачној инфраструктури (сервери, умрежавање, складиштење) инструкције за стварање нових домаћина, ВМ, виртуелних радних површина

Градивни блокови

- У општем случају је тешко обезбедити тачан дупликат основне машине
  - Нарочито ако је на процесору доступан само дуални-мод рада (кориснички и привилеговани)
  - Али с временом постаје све лакше како се побољшавају процесори функције и подршка за ВММ
  - Већина ВММ имплементира виртуелни процесор (*VCPU*) да представи стање процесора по домаћину онако како гост верује да јесте
  - Када се гостујући контекст ВММ пребаци на процесору, информације из *VCPU* се учитавају и чувају

Градивни блокови

- ОС користи режим језгра за заштиту од корисничког режима.
  - Системски позиви (привилеговане инструкције) генеришу прекид (софтверски прекид) који покреће прелазак у режим језгра
  - Ови позиви покрећу осетљиве инструкције (У/И, ММУ контрола, итд.) које само језгро може да извршава

Прекини и емулирај - *Trap and Emulate*

- Гост се извршава у корисничком режиму
- Језгро домаћина се извршава у режиму језгра
- Потребна су два режима за ВМ госта
  - режим виртуелног корисника и
  - режим виртуелног језгра
- Обоје раде у правом корисничком режиму
- Није сигурно дозволити да и језгро госта извршава у режиму језгра
- Акције у госту које обично узрокују прелазак у режим језгра морају да изазову прелазак у режим виртуелног језгра

Прекини и емулирај

Како се пребацује из режима виртуелног корисника у режим виртуелног језгра?

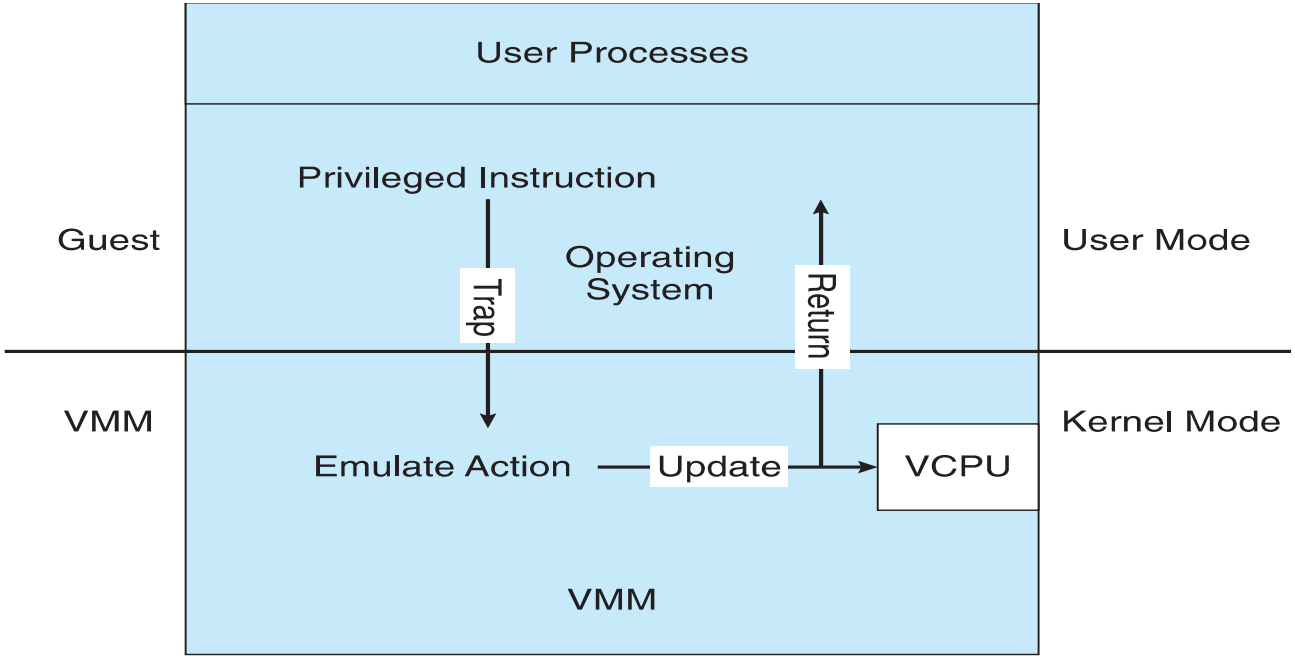
1. Покушај привилеговане инструкције у корисничком режиму изазива грешку -> прекид
2. ВММ стиче контролу, анализира грешку и извршава операцију као покушај госта
3. Враћа контролу госту у корисничком режиму

- Познат као прекидање и емулирање - ***Trap and Emulate***
- Већина програма за виртуелизацију ово барем делимично користи

Прекини и емулирај

- У корисничком режиму госту ради истом брзином као да није гост
- У режиму језгра, привилегованом режиму, гост ради спорије због прекидања и емулирања. Ово је изражено у случају већег броја гостујућих система.
- Неки процесори имају хардверску подршку, посебан режим за побољшање перформанси виртуелизације

Прекини и емулирај



Прекини и емулирај – проблеми код *x86*

- Неки процесори немају чисто раздвајање привилегованих и непривилегованих инструкција
- Ранији Интел x86 су међу њима
- Најранији Интелови процесори пројектовани су за рачунања
- Компатибилност уназад значи да је тешко побољшати
- Пример за ово је инструкција за Интел x86 *popf*
  - Учитава у програмску статусну реч индикаторе са стека
  - Ако је процесор у привилегованом режиму
    -> све индикаторе замењене
  - Ако је процесор у корисничком режиму
    -> само аритметичке индикаторе -> Нема прекида!

Списак инструкција које могу изазвати проблем

Problem category	Problem 80x86 instructions
Access sensitive registers without trapping when running in user mode	Store global descriptor table register (SGDT)
	Store local descriptor table register (SLDT)
	Store interrupt descriptor table register (SIDT)
	Store machine status word (SMSW)
	Push flags (PUSHF, PUSHFD)
	Pop flags (POPF, POPFD)

Списак инструкција које могу изазвати проблем

Problem category	Problem 80x86 instructions
When accessing virtual memory mechanisms in user mode, instructions fail the 80x86 protection checks	Load access rights from segment descriptor (LAR)
	Load segment limit from segment descriptor (LSL)
	Verify if segment descriptor is readable (VERR)
	Verify if segment descriptor is readable (VERR)
	Verify if segment descriptor is writable (VERW)
	Pop to segment register (POP CS, POP SS, …)
	Push segment register (PUSH CS, PUSH SS, …)
	Far call to different privilege level (CALL)
	Far return to different privilege level (RET)
	Far jump to different privilege level (JMP)
	Software interrupt (INT)
	Store segment selector register (STR)
	Move to/from segment registers (MOVE)

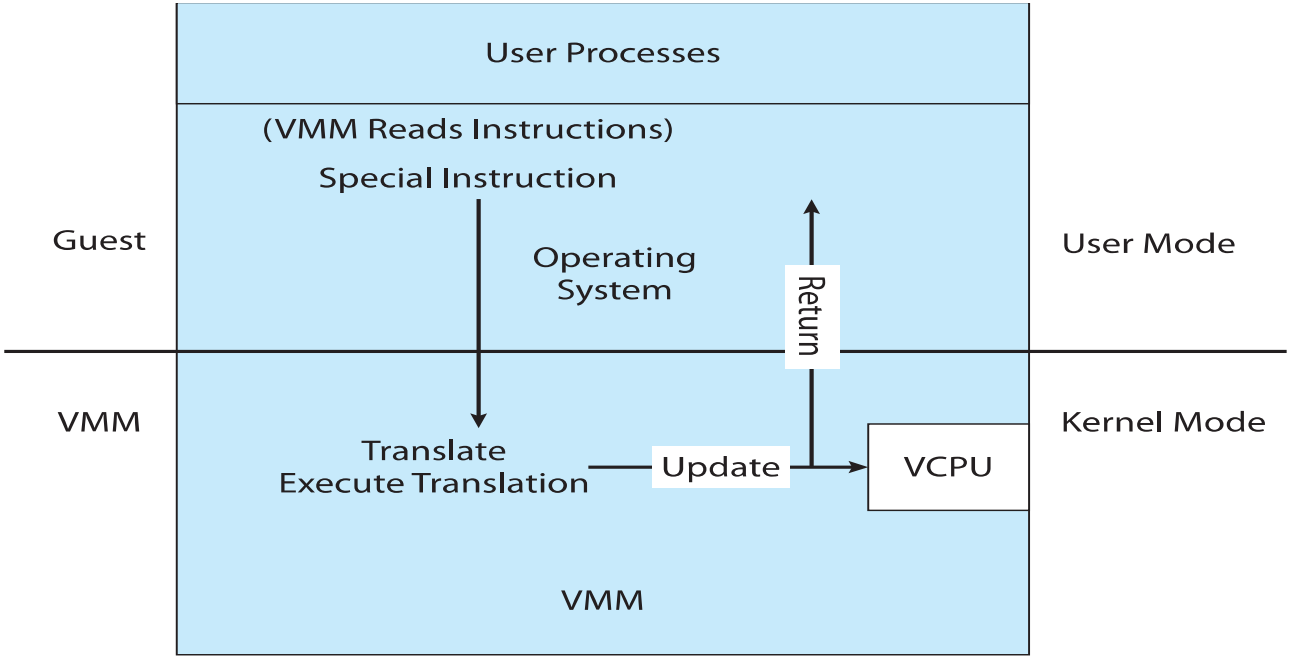
Бинарно превођење - *Binary Translation*

- Бинарни превод решава претходни проблем
- Једноставна основа али врло сложена примена
- Ако је гост корисничком режиму:
  - гост може да извршава инструкције неометано
- Ако је гост у режиму језгра (гост верује да је у режиму језгра):
  - ВММ проверава сваку инструкцију коју ће гост извршити (проверава неколико наредних инструкција)
  - Непроблематичне инструкције се извршавају неометано
  - Проблематичне инструкције се преводе у нови скуп инструкција које раде истоветан посао (на пример постављање индикатора)

Бинарно превођење

- Имплементирано превођењем кода унутар ВММ
- ВММ на захтев динамички чита инструкције госта и генерише бинарни код који се извршава уместо оригиналног кода
- Перформансе ове методе биле би лоше без оптимизација
- За убрзавање се користи кеширање (*VMware*)
  - Превести једном користити више пута
    Када гост изврши код који садржи спорне инструкције преусмерити га на извршавање већ преведеног кода а не на поново превођење
    (на пример покретање Виндоуз Икспе система је проузроковало 950.000 превода што је 5% успоравања)

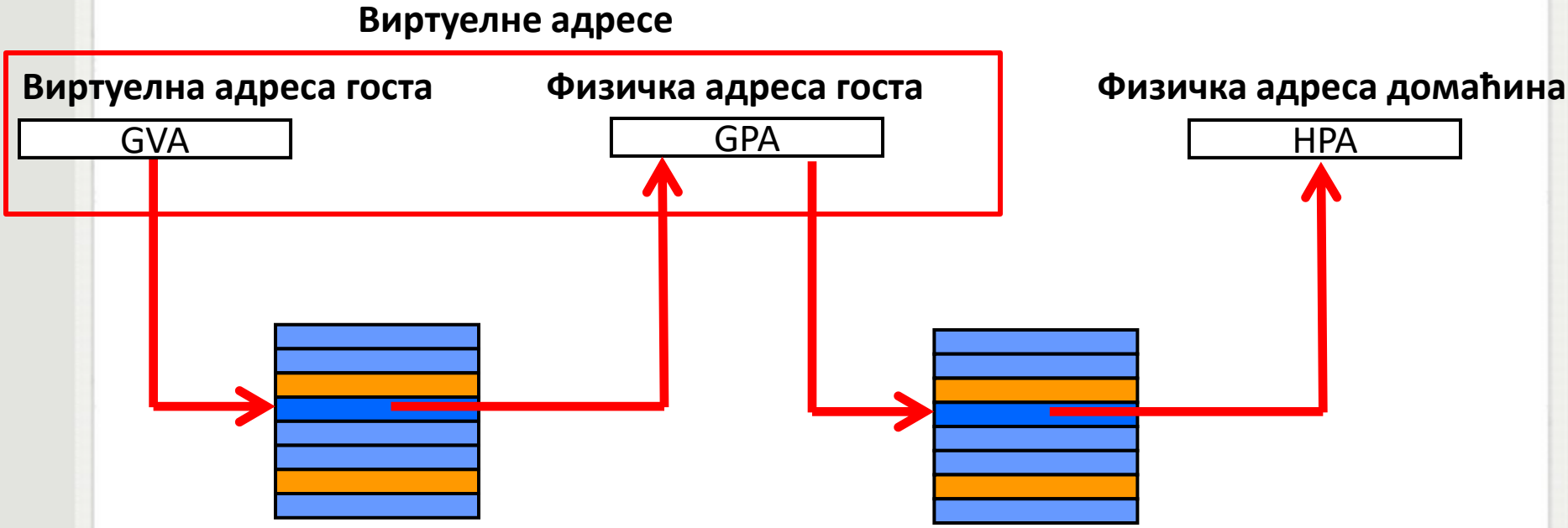
Бинарно превођење



Виртуелизација виртуелне меморије

- Код виртуелне меморије:
  - ОС прати пресликавање страница виртуелне меморије у странице физичке меморије
  - ОС попуњава табеле страница, а затим ажурира регистар странице (прекид)
- Како ВММ може задржати стање табеле страница како за госте који верују да контролишу табеле страница, тако и за ВММ који контролише табеле?

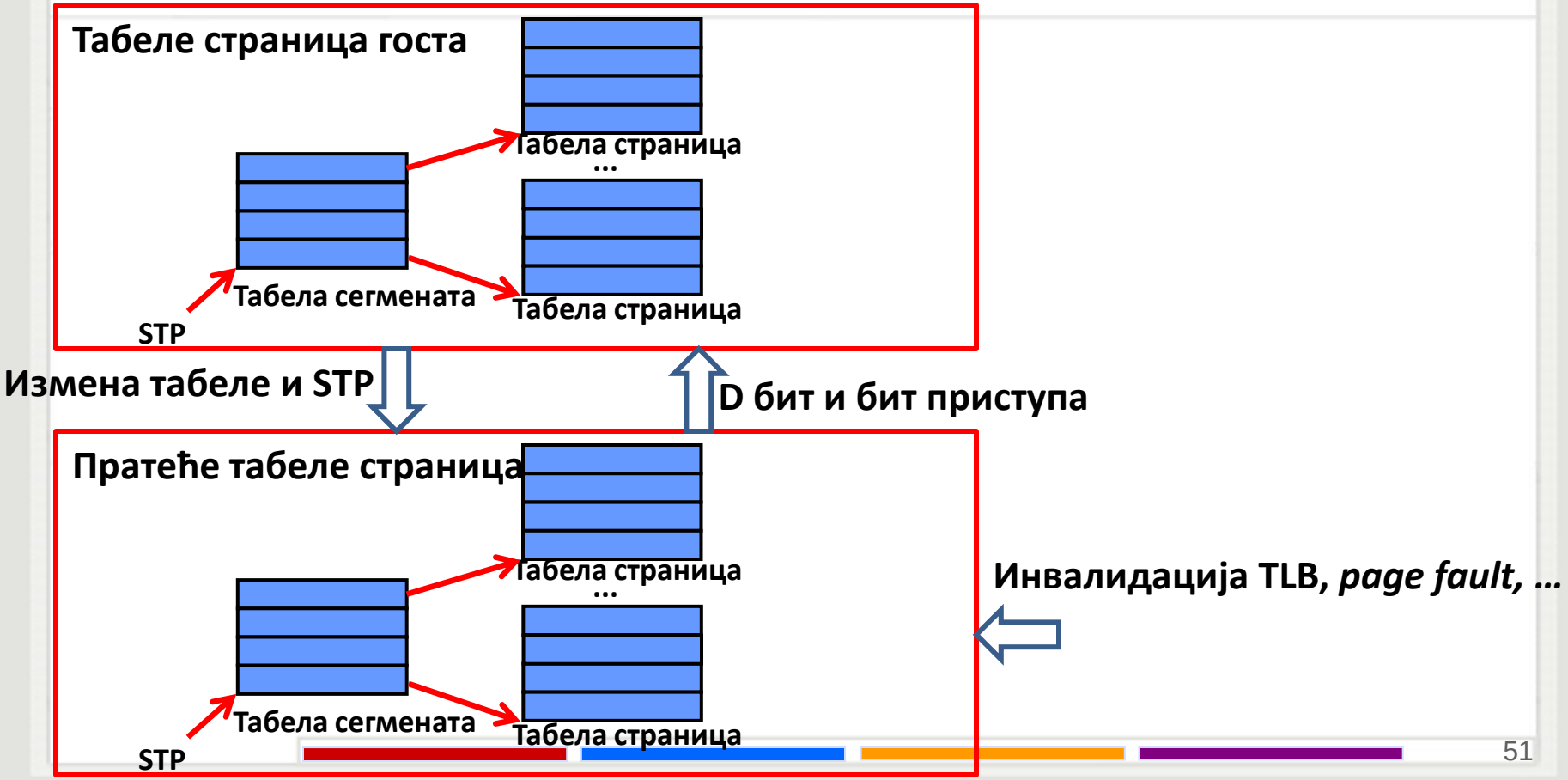
Виртуелизација виртуелне меморије



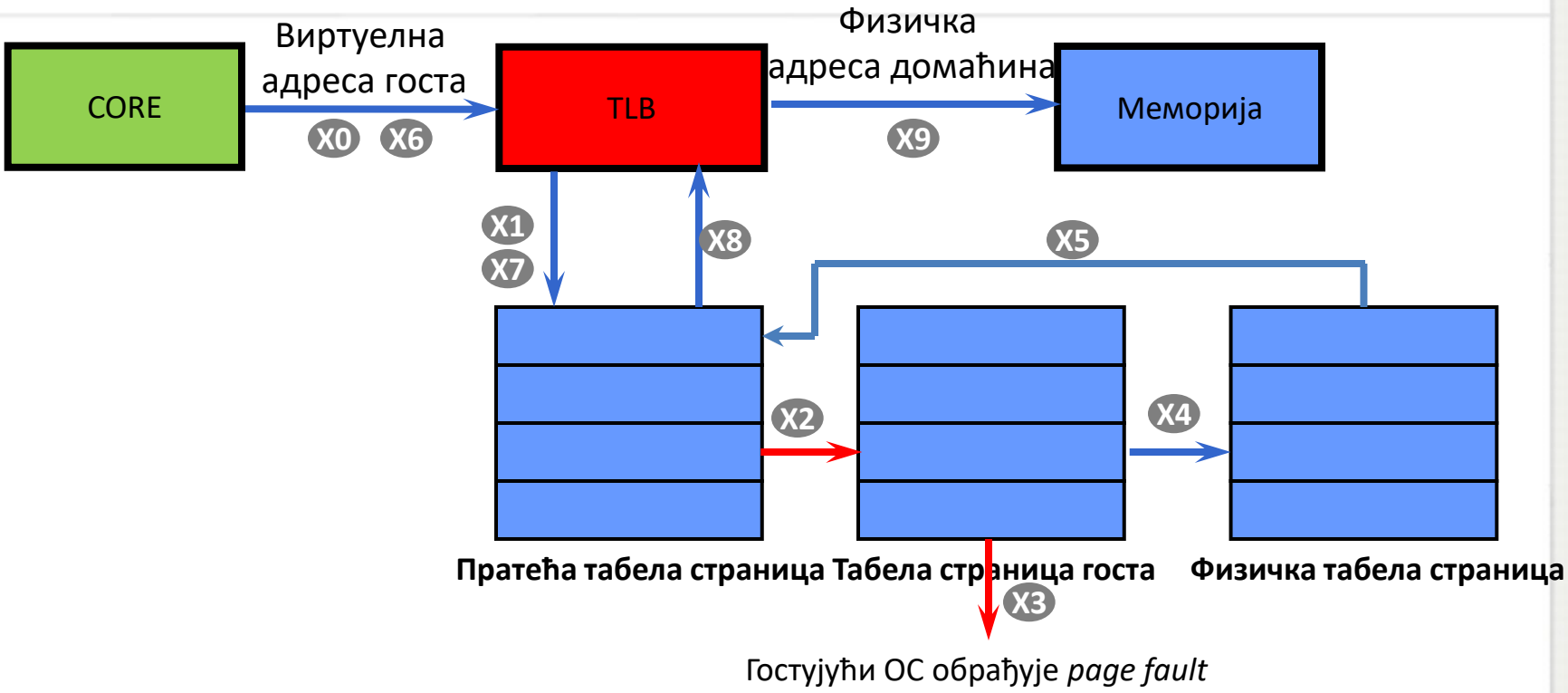
Виртуелизација виртуелне меморије

- Софтверска технике засноване на одржавају конзистентне верзије пратеће табеле страница (*shadow page table*) неке табеле страница госта (*guest page table*).
- Када је гост активан, ВММ присиљава процесор да користи пратећу табелу страница за превођења адреса.

Виртуелизација виртуелне меморије



Виртуелизација виртуелне меморије



Виртуелизација виртуелне меморије

- Пратећа табела страница није видљив госту.
- Да би одржао конзистентност ВММ мора да прати стање табеле страница госта и *STP* регистра.
- То укључује:
  - модификације које чини гост приликом додавања или уклањање превођења,
  - грешке приликом превођења адресе (*page fault*),
  - бите приступа и запрљаности,
  - инвалидацију *TLB* јединице
  - у случају симетричних мултипроцесора (*SMP*) и доследност превођења адреса на процесорима.

Виртуелизација виртуелне меморије

- Примена у STP регистра генерише прекид.
- Грешке приликом превођења адресе је прекид.
- Промене у табелама страница **не генеришу прекид**!

Виртуелизација виртуелне меморије

- Једно решење: Означити табеле страница госта само за читање (*write-protecting*). Када гост покуша да приступи, дода или обрише улаз, у табелу догоди се прекид услед права приступа који обради ВММ. У прекидној рутини се синхронизују табеле.

Виртуелизација виртуелне меморије

- Друго решење: Заснива се грешкама приликом превођења и одржавања конзистентности *TLB* јединицом. У овој техници, која се понекад назива и виртуелни *TLB*, ВММ омогућава госту да дода нове преводе у своју табелу страница без пресретања тих операција. Када гост приступи инструкцији или подацима који се односе на дато превођење деси се прекид усред грешка приликом превођења (*page fault*) јер тај превод још увек није присутан у пратећој табели. Овај прекид омогућава да ВММ прегледа табелу госта и да дода превод који недостаје у пратећој табели. Слично томе, када гост уклони превод, извршава инвалидацију *TLB* јединице (*INVLPG*) да би онемогућио тај превод. ВММ пресреће ову операцију; затим уклања одговарајући превод у пратећој табели и извршава инвалидацију за уклоњени превод.

Виртуелизација виртуелне меморије

- Обе технике резултирају великим бројем прекида (*page fault*).
  - Прекиди које је проузроковао гост. Настају услед нормалног понашања госта када се приступа страницама које је гостујући оперативни систем сачувао на диску. Хипервизор мора пресрести ове прекиде, анализирати и затим осликати у госту, што је значајан додатни посао у поређењу са оригиналним страничењем.
  - Прекиди које је проузроковао хипервизор. Прекиде због пратећих страница.
- Разликовање ова два типа прекида доводи до значајног додатног посла за софтвер.

Виртуелизација виртуелне меморије

- Треће решење: Паравиртуализовани ОС. Будући да је ОС модификован да би узео у обзир ВММ, ажурирања табеле страница могу бити праћена позивима ка ВММ о променама.

Виртуеизација уређаја

- Хипервизор виртуелизује физички хардвер и сваком госту представља стандардизовани скуп виртуелних уређаја
- Сваки гост мисли да приступа физичком уређају
- Виртуелизација уређаја укључује управљање усмеравањем У/И захтева између виртуелних уређаја и дељеног физичког хардвера
- Ови виртуелни уређаји ефикасно опонашају добро познати хардвер (са управљачким програмима - драјвера) и преводе захтеве виртуелне машине на системски хардвер
- Ова стандардизација на нивоу управљачких програма такође помаже у стандардизацији и преносивости виртуелних машина на платформама, јер су све виртуелне машине конфигурисане да раде на истом виртуелном хардверу, без обзира на стварни физички хардвер у систему

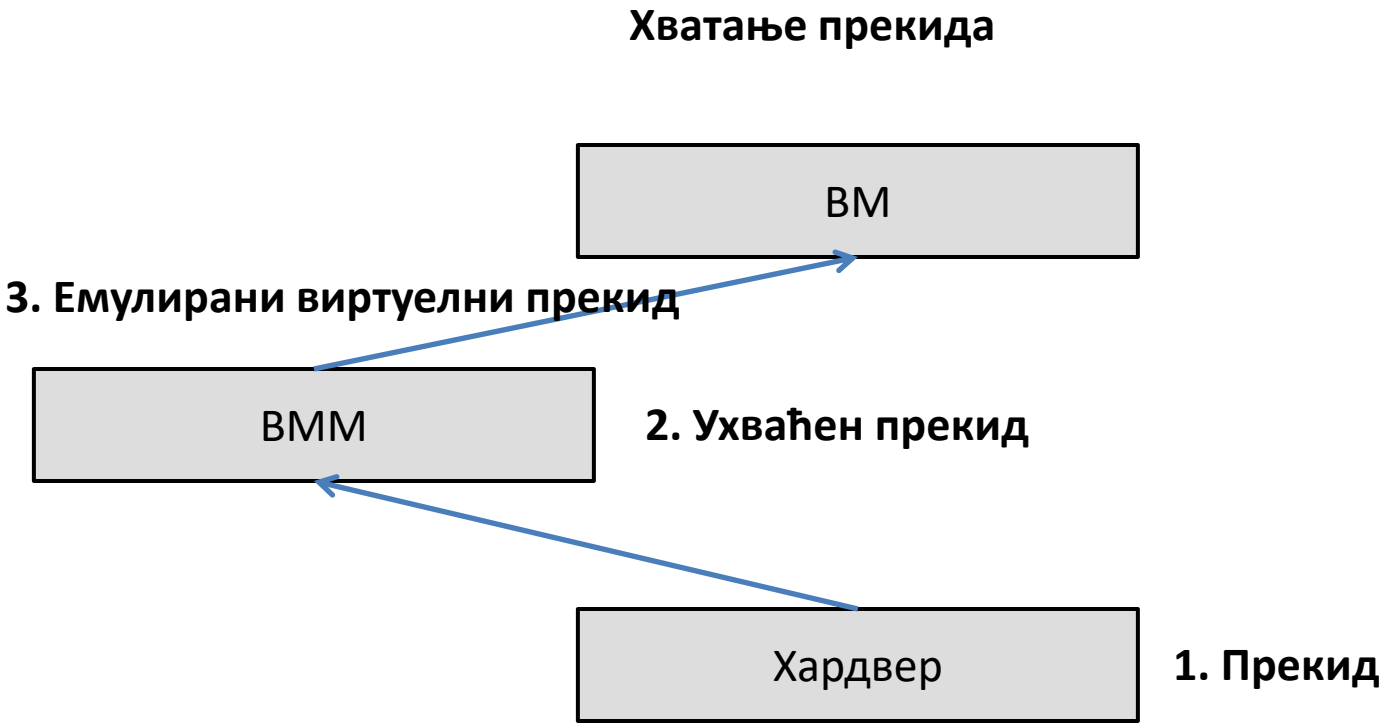
Виртуеизација уређаја – диск

- Сваки гостујући оперативни систем има свој део диска.
  - Типично се имплементира као датотека или регион на диску
  - Сваки гост мисли да приступа физичком диску
  - Апстракције виртуелног диска
  - ВММ мора преводити гостујућу адресу на виртуелном диску у физичку адресу у додељеном региону на диску
  - Стварне партиције могу се креирати на диску и доделити гостима, или читав виртуелни диск може бити датотака на диску
  - Ово се може пребацивати између сервера

Виртуеизација уређаја – мрежни прикључак

- Физичка машина има мрежну картицу која се користи за мрежни У/И
- Гости деле ову физичку мрежну картицу
- Сваком госту је додељена логичка мрежне картица помоћу које врши мрежни У/И
- Мрежне пакете које шаљу гост хипервизор мултиплексира на стварну физичку мрежну картицу
- Слично томе, када пакети стигну на стварну мрежну картицу, хипервизор утврђује за коју гостујући мрежу су намењени и испоручује их у складу с тим

Виртуеизација прекида



Проширења архитектуре рачунара

- Циљеви:
  - Користити угњеждене табеле страница уместо сакривених табела страница
  - Избегнути пражњење *TLB* јединице
  - Дозволити оперативном систему госта да опслужује прекиде
  - Дозволити уређајима да користе DMA за пренос података
  - Дозволити програмима да раде за шифрираним деловима кода и података

Проширења - *x86*

Интел - ***VT-x*** (*Vanderpool*) 2005

АМД - ***AMD-V*** (*Secure Virtual Machine*) 2006

Нису компатибилни!

- Процесори са додатим инструкцијама уклањају потребу за бинарним превођењем
- Дефинисано је више режима рада процесора - гост и домаћин
- ВММ може омогућити режим домаћина, дефинисати карактеристике сваког госта, пребацити се у режим госта и пребаците госте на одговарајући процесор
- У режиму госта, гостујући ОС мисли да ради изворно, види уређаје (онако како је то дефинисао ВММ)
  - Приступ виртуелизованом уређају
  - Бољa поставка инструкција које изазивају прекид
  - Процесор одржава VCPU контекст и мења га по потреби

Проширења - *x86-32* и *x86-64*

- Хардверски модови за госте како би се редуковало прекидање. Детаљи:
  - *VMX* мод рада за гостујући оперативни систем
  - *VMCB* – контролни блок виртуелне машине
  - *vmrun* инструкција за улазак у *VMX* мод
  - Више инструкција и догађаја омогућава излазак из *VMX* мода
  - Контролни бити у VMCB могу променити начин излази из *VMX*
- Виртуелизација прекида
  - Интел - *APIC virtualization (APICv)*[
  - АМД - *Advanced Virtual Interrupt Controller (AVIC)*

Дозвољен потпуно коришћење технике ***trap-and-emulate***!

Проширења - *x86-32* и *x86-64*

- Стање госта, виртуелне машине, се чува у структури *VMCB* (*Virtual Machine Control Block*)
- Величине је 4 KB
- Садржи стање и информације које инструкције и догађаје да пресреће (регистри, услови за излазак из мода, пратеће прекиде након *STI* инструкције, ...)
- Након преласка у режим домаћина хипервизор може проверити детаљно стање машине засновано не овој структури

Проширења - *x86-32* и *x86-64*

- Хипервизор може контролисати који догађаји у режиму језгра (*ring 0*) проузрокују прекиде
- Проблематичне инструкције у режиму језгра (*popf, pop CS, ...*) такође генеришу прекид
- Изузеци се могу искључити како би допринело перформансама, неки изузеци су селективни (промена бита мода рада у контролном регистру процесора)
- Могуће је пресретнути и *vmrun* како би се омогућило и гнежђење виртуелних машина

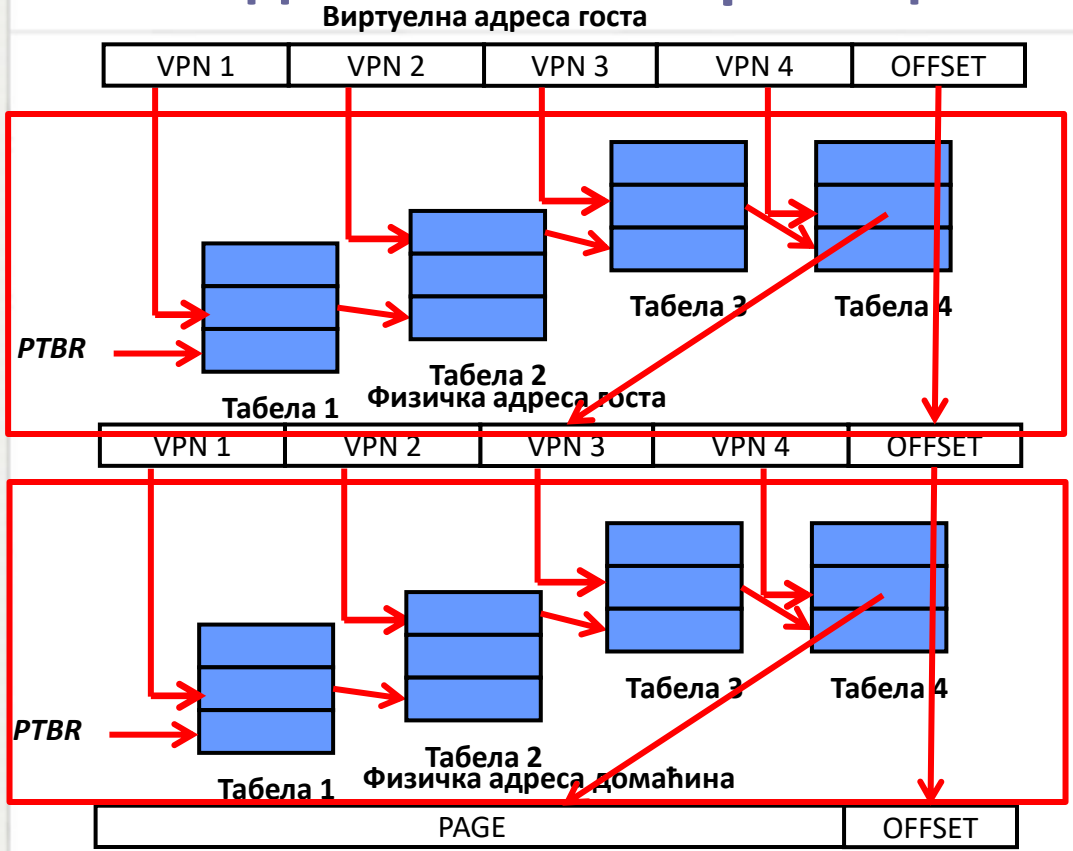
Проширења - *x86-32* и *x86-64*

- Проширен је таг у *TLB* јединици (ВММ и одвојено за сваки адресни простор ВМ) како би се избегло пражњење јединице приликом преласка из једног мода у други
- Странични реални мод рада је додат како би се дозволило ефикасно симулирање реалног мода

Проширења - *x86-32* и *x86-64*

- Додата је подршка за угњеждене табеле
- Адресе генерисане у ВМ су у линеарном адресном простору госта, које се онда даље преводе у линеарни адресни простор домаћина
- Овај механизам елиминише потребу за пратећим страницама
- Оперативни систем госта може одржавати своје табеле страница без потребе за пресретањем ових захтева

Угњеждене табеле страница



Табеле страница госта

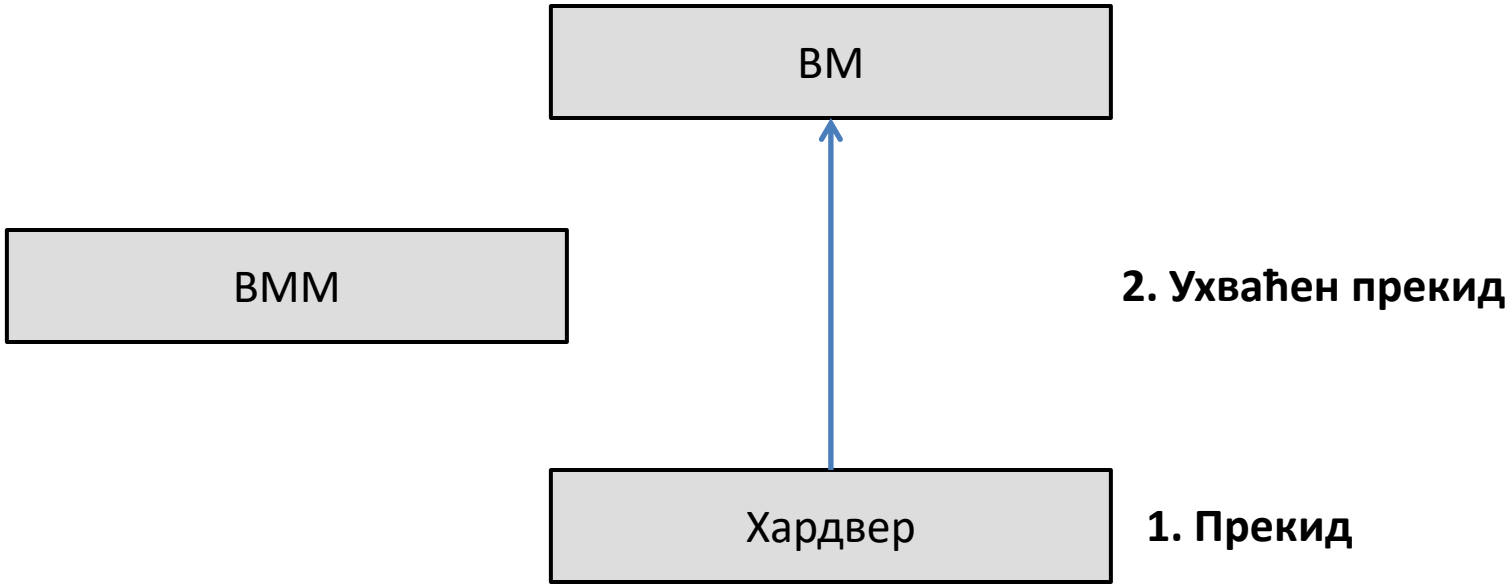
Угњеждена табела страница

Проширења - *x86-32* и *x86-64*

- Могуће је прослеђивање захтева за прекид *IRQ* до назначеног госта, пресретнути захтеве за прекидом госта и позвати симулирани прекид на госту
- Могуће је пресретнути операције са улазно излазним адресним простором (*I/O ports*) код госта селективно (и дозволити да се некима директно приступи)
- Додата је могућност ограничења приступа хардверских уређаја одређеним регионима физичке меморије да би се обезбедила изолација виртуелних машина које имају директан приступ хардверским уређајима (*IOMMU*)

Виртуеизација прекида

Хватање прекида у посебном моду



Питања?

Електротехнички Факултет
Универзитет у Београду