Hardwarové principy virtualizace

Hardware virtualization principles

Kateřina Kollariková

Bakalářská práce
2011

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Kateřina KOLLARIKOVÁ**
Osobní číslo:	**A07549**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Hardwarové principy virtualizace**

Zásady pro vypracování:

1. **Popište princip virtualizace obecně, její možnosti a výhody.**
2. **Zaměřte se na hardwarovou stránku problematiky.**
3. **Popište základní principy technologie IVT (Intel Virtualization Technology).**
4. **Popište základní principy technologie AMD-V (AMD virtualization).**
5. **Porovnejte uvedené technologie, popište možnosti využití.**
6. **Vytvořte názorné vizualizace základních principů virtualizace.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KRÁL, Jan; KRAHULEC, Lukáš. WikiHosting [online]. 5.12.2008 [cit. 2011-01-31]. PAP_Virtualizace_referát_(Krahulec_Král).pdf. Dostupné z WWW: http://wh.cs.vsb.cz/mil051/images/f/f5/PAP_Virtualizace_refer%C3%A1t_%28Krahulec_Kr%C3%A1l%29.pdf.**
2. **HORÁK, Jaroslav . Hardware : učebnice pro pokročilé. 4. aktualizované vydání. Brno : Computer Press, a.s., 2007 . 360 s. ISBN 978-80-251-1741-5.**
3. **Intel [online]. Intel Corporation : Copyright, 2007-2010 [cit. 2010-10-31]. Dostupné z WWW: http://www.intel.com/Assets/PDF/manual/253669.pdf.**
4. **GOWDA, Bhaskar . Intel [online]. 3.6.2009 [cit. 2011-01-31]. The Server Room Blog. Dostupné z WWW: http://communities.intel.com/community/openportit/server/blog/2009/06/02/a-peek-into-extended-page-tables?wapkw=%28Extended+page+tables%29%29.**
5. **AMD [online]. 2008 [cit. 2011-01-31]. AMD Virtualization (AMD-V?) Technology . Dostupné z WWW: http://developer.amd.com/assets/NPT-WP-1%201-final-TM.pdf.**

Vedoucí bakalářské práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **25. února 2011**

Termín odevzdání bakalářské práce: **7. června 2011**

Ve Zlíně dne 25. února 2011

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Tato bakalářská práce se zabývá hardwarovou virtualizací. V úvodu práce je definován pojem virtualizace, základní typy virtualizace a možnosti jejího využití. V další části jsou podrobně vysvětleny základní principy technologie VT-x od firmy Intel a technologie SVM od firmy AMD, které mají ukázat, co hardwarová virtualizace přináší. V závěru teoretické části jsou popsány hlavní rozdíly obou uvedených technologií.

V praktické části jsou vizualizovány základní prvky hardwarové virtualizace v PowerPointu.

Klíčová slova: virtualizace, hypervizor, virtuální stroj, procesor, technologie VT-x, technologie SVM, virtualizace paměti

ABSTRACT

This bachelor work engages in a hardware virtualization. In the introduction is defined the conception of virtualization the elementary types and some possibility it's employs. In the next section there are explained the fundamental principles of technology VT-x from Intel company and the technology SVM from AMD company to show what hardware virtualization provides. The main diferences of these technologies are closely described at the end of the theoretical part.

The basic elements of hardware virtualization are visualizated in the practical part in PowerPoint.

Keywords: virtualization, hypervisor, virtual machine, processor, technology VT-x, technology SVM, memory virtualization

Poděkování

Ráda bych poděkovala panu Doc. Ing. Martinu Syslovi, Ph.D. vedoucímu mé práce za cenné rady a připomínky, které mi poskytl v průběhu vypracování této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně dne 31. 5. 2011 …….

podpis diplomanta

OBSAH

ÚVOD

Tato práce se zabývá hardwarovou virtualizací. Virtualizace patří v dnešní době mezi moderní technologie ve světě IT. Umožňuje na jednom fyzickém počítači spustit najednou více operačních systémů – virtuálních strojů. Můžeme například konsolidovat servery, aniž bychom museli kupovat další hardware nebo díky virtualizaci můžeme testovat a ladit programy bez nutnosti instalovat další testovací verze operačního systému.

Hlavním prvkem pro hardwarovou virtualizaci je hypervizor někdy označován jako VMM *Virtual Machine Monitor*, který se stará o to, aby jednotlivá jádra nesahala tam, kam nemají. Hardwarová podpora virtualizace převádí některé úkoly VMM do režie hardware. To zajišťuje vyšší ochranu a izolaci prostředků hostovaných systémů. Návrh hypervizoru je mnohem jednodušší, obsahuje méně kódu, a proto je robustnější a bezpečnější.

V této práci je představen průběh hardwarové virtualizace dvou největších společností. Společnost Intel pod označením VT-x uvedla svou technologii v procesoru v roce 2005 a rok později v roce 2006 společnost AMD představila technologii SVM *Secure Virtual Machine*. Obě tyto technologie zavádí novou úroveň ochrany ring -1, která je nadřazena všem úrovním a je vyhrazena pro běh VMM *Virtual Machine Monitor*.

První generace uvedených technologií neobsahovala speciální podporu pro virtualizaci paměti. Společnost AMD uvedla tuto techniku v roce 2007 pod označením NPT *Nested Page Table* a Intel v roce 2009 pod označením EPT *Extended Page Table*, které odstranily nedostatek vyšší režie.

V závěru teoretické části jsem uvedla porovnání obou uvedených technologií.

Praktická část je vytvořena v aplikaci PowerPoint, kde jsem vizualizovala techniky hardwarové virtualizace u technologie VT-x od firmy Intel.

I. TEORETICKÁ ČÁST

1 VIRTUALIZACE

Umožňuje provozovat v hostitelském operačním systému tzv. virtuální stroje. Hardwarová virtualizace dovoluje na jednom počítači spustit více systémů (virtuálních strojů) najednou, přičemž přepínání mezi nimi je díky hardwarové podpoře poměrně rychlé a efektivní. Hardwarová implementace virtualizace zjednodušuje návrh software, snižuje režii, zvyšuje rovněž bezpečnost jednotlivých virtuálních strojů jejich zvýšenou izolací.

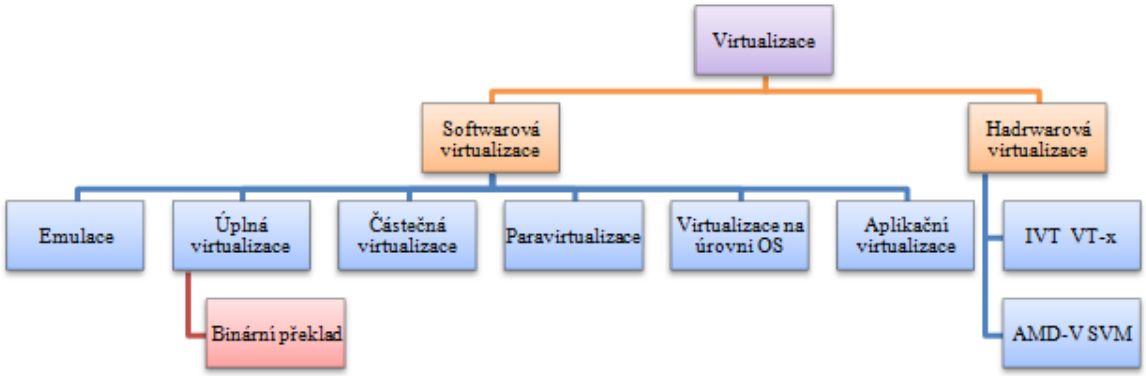
Virtuální stroje můžeme například použít pro:

- Testování a ladění programů bez nutnosti instalovat testovací verze operačního systému.
- Prezentaci a výuku, kdy na jednom počítači můžeme předvádět několik operačních systémů (např. Linux a Windows Server) zároveň.
- Simulaci provozních stavů.
- Konsolidaci serverů bez nutnosti kupovat další hardware – na jednom počítači pracuje několik serverových operačních systémů.

Společnosti, které se v současnosti zabývají hardwarovou virtualizací:

- Intel zavádí postupně u nových procesorů technologii původně označovanou Vanderpool, jejíž oficiální název je *Intel Virtualization Technology.*

- AMD vyvíjela virtualizační technologii pod názvem Pacifica, její oficiální název zní *AMD Virtualization.* [3, 10]



Obr. 1 *Schéma technik virtualizace* [14, 15]

Dále budou představeny techniky Úplné virtualizace, v praxi nejčastěji je využívaná Binární překlad, dále potom Paravirtualizace a techniky Hardwarové virtualizace – od IVT *Intel Virtualization Technology* VT-x a od AMD-V *AMD Virtualization* SVM *Secury Virtual Machine* s podporou hypervizoru. [20]

1.1 Hypervizor

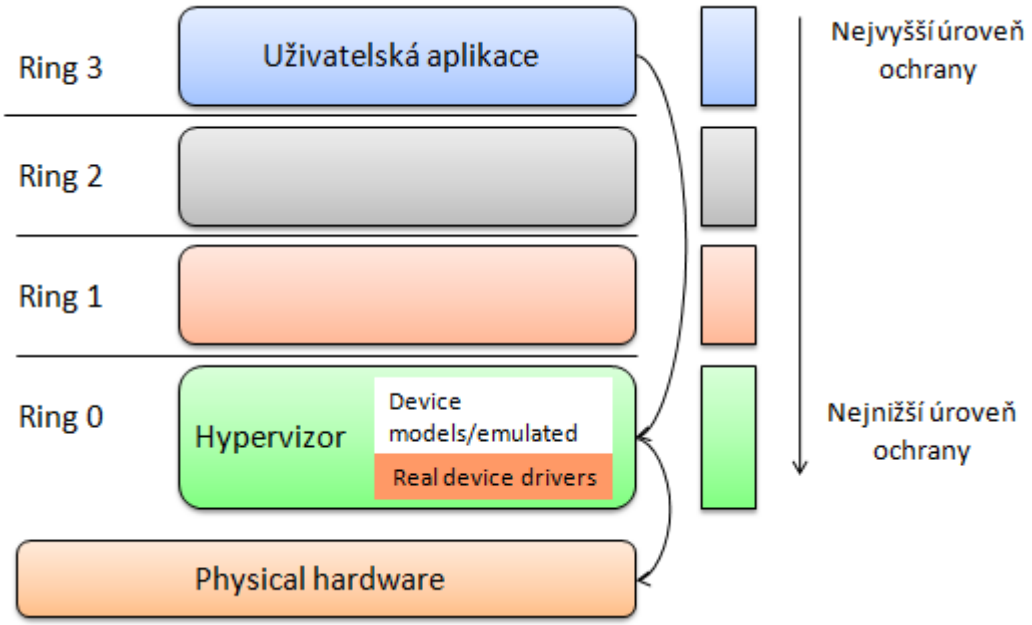
Ještě než bude provedeno další rozdělení, bude představen hlavní prvek virtualizace a jaké jsou nutné změny v rozložení SW.

Hlavní vlastností virtualizace je hlavně obsluha CPU a hlavně jeho *privilegovaných instrukcí* (neprivilegování instrukcí nebo jejich obsluha), *obsluha paměti* (stránkování, přidělování paměti) a *virtualizace vstupně / výstupních zařízení* (obsluha požadavků pro přidělení disku, sítových rozhraní).

Z důvodu ochrany procesor v privilegovaném režimu zavádí ochranné úrovně – ring. Celkem 4 ringy rozdělují privilegia k přístupu k samotnému HW, kde nejnižší ring 0 má nejvyšší pravomoci a kód vykonávaný v tomto ringu může pracovat se samotným HW – tedy může vykonávat privilegované instrukce. V tomto ringu je tedy umístěno jádro operačního systému. Uživatelské programy by neměly mít právo sahat přímo na hardware, a proto běží až v ringu 3. Ringy 1 a 2 jsou nepoužívané.

Hypervizor je nejzákladnějším prvkem hardwarové virtualizace. Jelikož se obvykle virtualizuje více operačních systémů, je tedy nutné je oddělit tak, aby se vzájemně nepoškozovaly. Jejich jádro se přesouvá do ring 1 a do ring 0. Zde se zavádí hypervizor (nebo taky obecně – *Virtual Machine Monitor*), který se stará o to, aby jednotlivá jádra

nesahala, kam nemají, poskytuje jim rozhraní pro volání (paravirtualizace), stará se o binární překlad instrukcí atd. [1]



Obr. 2 *Architektura hypervizoru*

1.2 Technika úplné virtualizace bez podpory procesoru

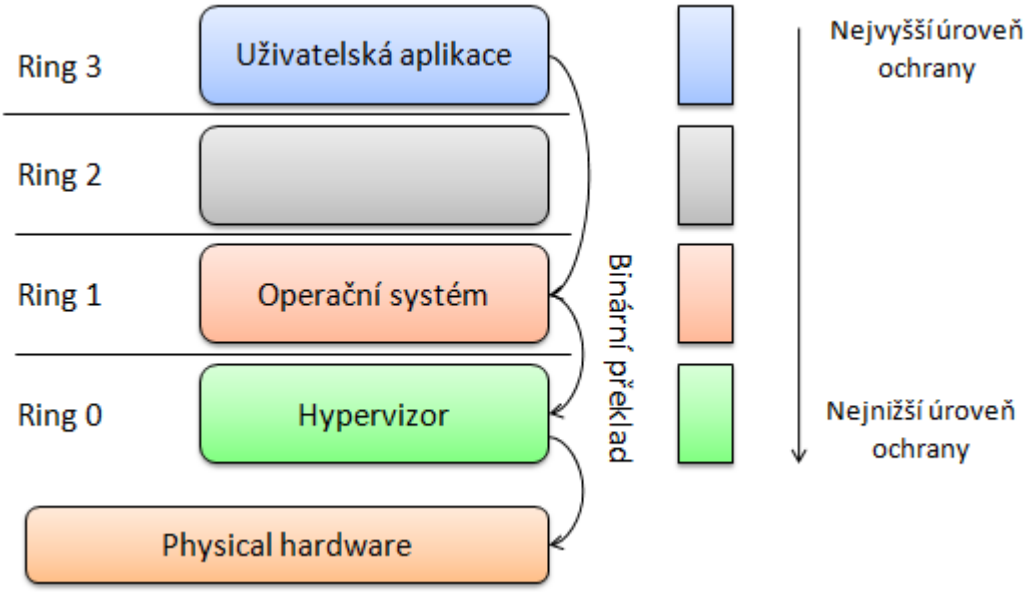
1.2.1 Binární překlad

Je nejčastější technikou pro úplnou virtualizaci. Tato technika překládá binární kód, který chce OS hosta vykonat za běhu na upravený (bezpečný) kód. V podstatě programy běžící v uživatelském módu ring 3 vykonávají svůj kód přímo, a jakmile je požadováno vykonání nebezpečného kódu (privilegované instrukce), pak je VMM *Virtual Machine Monitor* „požádán“ o překlad na bezpečné volání. Pokud například OS požaduje přístup k fyzickému hardware (např. HDD), pak je tento požadavek přeložen jako požadavek k přístupu do virtuálního hardware (vyhrazený oddíl na disku).

Překladem vlastně přesunuje kód, který je spustitelný z ringu 0 na kód, který poběží až v ringu 3.

Binární překlad nemá za cíl optimalizovat, ale pouze překládat privilegované instrukce na bezpečné. Zároveň ukládá přeložené instrukce do cache, takže například při instrukci cyklu není kód překládán znovu, ale je vybrán z cache.

Jistou nevýhodou překladu je určitě rychlost, jelikož se musí každá neprivilegovaná operace přeložit a vzniká veliká režie, ale na druhou stranu díky cache a ukládáním přeloženého kódu doba překladu postupně klesá. [1]



Obr. 3 *Binární překlad* [20]

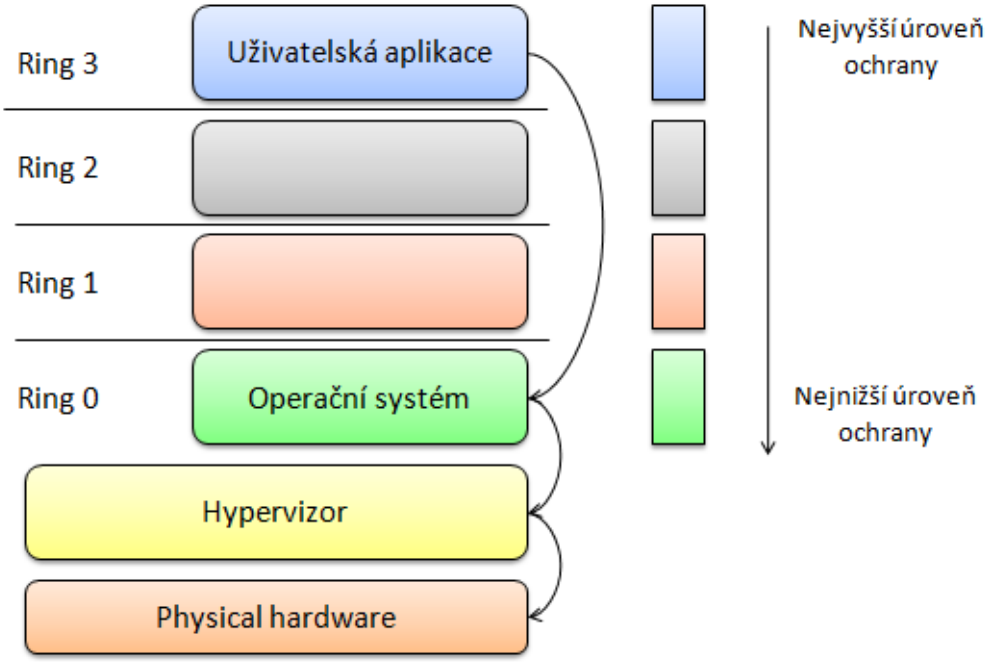
1.3 Paravirtualizace

Při paravirtualizaci se prostředí pro virtualizovaný stroj nevirtualizuje plně. V podstatě je to technika podobná binárnímu překladu, ale s tím rozdílem, že překlad probíhá už na úrovni zdrojového kódu. Není dále pak nutné překládat za běhu instrukce. Samozřejmostí je, že tato technika je použitelná jenom v tom případě, že máme přístup do zdrojového kódu systému pro provedení změn.

Hlavní úpravou je samozřejmě úprava kritických operací. Hostovaný systém je upraven na nová systémová volání, která jsou nahrazena tzv. hypercall voláním.

Velkou výhodou paravirtualizace je tedy značné urychlení a to hlavně odstraněním nutnosti překladu instrukcí, kdy se volá přímo hypervizor pomocí jeho rozhraní API *Application Programming Interface*. Odpadá tedy značná část nutné režie.

Nevýhodou je jistě nutnost zásahu (která neznamená přepis jádra), a tedy pro paravirtualizaci je možné využít pouze otevřené systémy. [1]



Obr. 4 *Paravirtualizace* [20]

1.4 Virtualizace s podporou procesoru

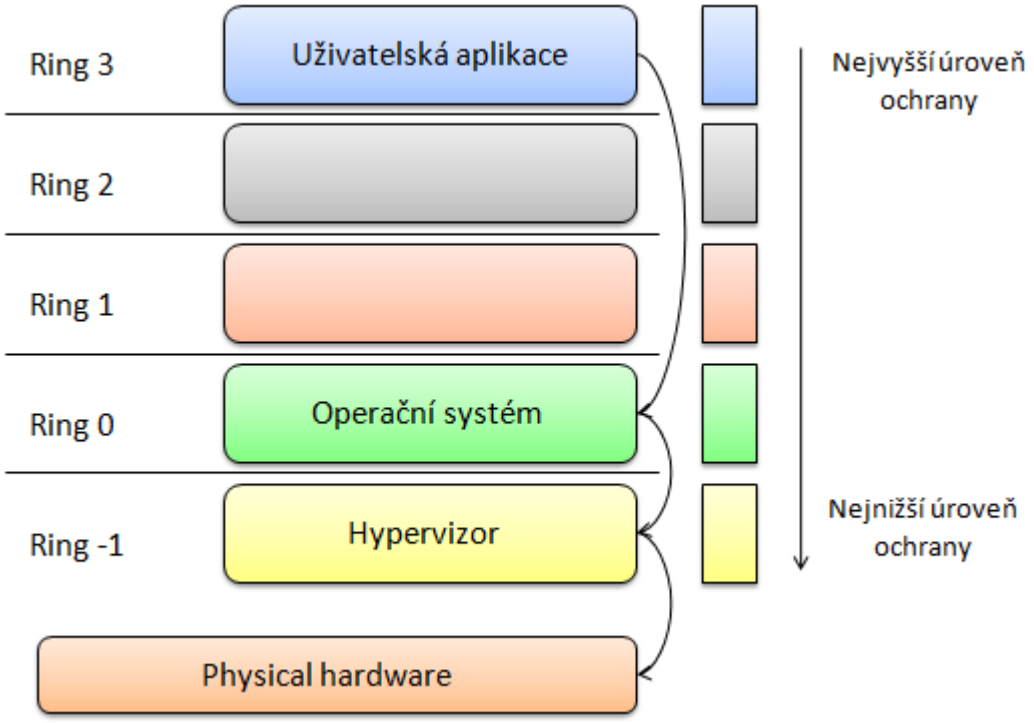
IVT a AMD-V nabídly hardwarovou podporu virtualizace. Základní myšlenkou hardwarové virtualizace je snažit se zachytit všechny výjimky procesoru a všechny privilegované instrukce vložením vynuceného přechodu z hostovaného OS do VMM.

1.4.1 Model hardwarově podpořené virtualizace

Jde o rozšíření možností procesoru tak, že přibývá další úroveň ochrany s ještě většími oprávněními „ring -1“, která je nadřazena všem úrovním. Na této úrovni přibývají speciální instrukce. Úroveň ring -1 je určen pro běh VMM. Díky tomu může OS hostovaného systému běžet na své určené úrovni ochrany, kterou je ring 0. Virtuální stroje tak pracují v prostředí, které se neliší od nativního.

Hostovaný OS poskytuje uživatelské aplikaci služby jádra a VMM zasáhne pouze, když systémové volání vyvolá kritické instrukce. To znamená, že nyní už systémové volání *systém call* automaticky neznamená zásah VMM. [1]

** Více v kapitole [3.1] Technologie VT-x*



Obr. 5 *Hardwarová podpora virtualizace procesory AMD SVM a Intel VT-x* [20]

1.5 Příklady použití virtualizace

1.5.1 Virtuální stroj

Je obraz počítače, který však existuje jen jako model uvnitř jiného počítače. Programy běžící uvnitř tohoto stroje se chovají, jako by běžely na skutečném počítači, ale přitom nemohou nijak ovlivnit zbytek „vnějšího“ počítače. Virtuální stroje se používají z bezpečnostních důvodů pro běh některých aplikací, aby se tyto aplikace nemohly navzájem žádným způsobem ohrozit, ani ovládnout hostitelský počítač, resp. jeho operační systém. Některé programovací jazyky byly navrženy přímo pro běh ve virtuálním stroji a zpravidla

se nepředpokládá, že by v nich napsané programy běžely přímo na hardwaru. Příkladem takových jazyků je Java či C++. [14]

1.5.2 Virtualizace úložiště

Určitě každý server, ale i některé osobní počítače využívají RAID *Redundant Array of Independent Disks* pole. Jedná se vlastně o metodu, jak koordinovat práci více pevných disků, které se tváří jako jediný. Tímto dosáhneme většího zabezpečení ztráty uložených dat způsobenou chybou na disku, ale i získáme navíc například urychlení čtení z disku. [14]

1.5.3 Virtuální paměť

Je způsob, jakým programy mohou pracovat s větším množstvím paměti, než počítač ve skutečnosti obsahuje, odděluje paměť jednotlivých procesů atd. [14]

2 PROCESOR

Hardwarová podpora virtualizace je založena na čipu. Nejprve bude představen, co je a jak funguje procesor.

Procesor CPU *Central Processing Unit* je „mozkem“ počítače, který zpracovává instrukce programů, kterými je řízen. Některé instrukce zpracovává sám, k provedení dalších instrukcí používá různé komponenty počítače (operační paměť, disky, sběrnice, displej, tiskárny…apod.)

Základní vlastností procesoru je tedy programovatelnost a integrace všech základních obvodů do jediného pouzdra.

Jádrem každého mikroprocesoru je logický obvod, který dokáže zpracovat sadu jednoduchých mikroinstrukcí. Mikroinstrukce jsou jen jednoduché příkazy. Každý mikroprocesor je navíc vybaven instrukční sadou, která programátorům poskytuje přívětivější prostředky pro napsání složitějšího aplikačního programu. Převod instrukční sady na mikroinstrukce, které je mikroprocesor schopen řešit, obstarává program napsaný v mikroinstrukcích. Ten je další podstatnou částí mikroprocesoru.

2.1 Činnost procesoru

2.1.1 Registry

Každý mikroprocesor obsahuje registry. Ty fungují jako vnitřní paměti mikroprocesorů, do kterých se ukládají momentálně zpracovávaná data. Počet registrů a jejich přesné použití se u jednotlivých mikroprocesorů liší. K specifikaci adres v paměti, na kterých se nacházejí zpracovávaná data, se používají různé způsoby adresovacích mechanismů. [2]

2.1.2 Systém přerušení

Přerušení je signál, který k mikroprocesoru vyšle některé hardwarové zařízení nebo program. Vysílatel signálu se tak snaží zabrat mikroprocesor pro sebe. Klasickým příkladem je stisk klávesy na klávesnici. Mikroprocesor musí přerušit svoji činnost a povel daný klávesou zpracovat.

Všechny moderní mikroprocesory mají vektorový systém přerušení. To znamená, že každé přerušení je identifikováno svým číslem. Na určitém místě v operační paměti je uložena tabulka vektorů přerušení. Vektor přerušení, identifikovaný právě číslem přerušení, ukazuje na adresu paměti, kde je uložen obslužný podprogram přerušení. N-té přerušení spustí (přes n-tý vektor přerušení) n-tý program, který zpracuje požadavek zdroje přerušení.
Před skokem na vektor přerušení uloží mikroprocesor svůj momentální stav do speciálního registru – zásobníku. To mu umožní vrátit se po zpracování přerušení k původní činnosti. Výhodou vektorového přerušovacího systému je možnost nahrazení obslužného programu přerušení programem vlastním.
Mikroprocesor musí obsahovat i mechanismus, kterým přerušení dočasně zakáže. [3]

2.1.3 Správa paměti

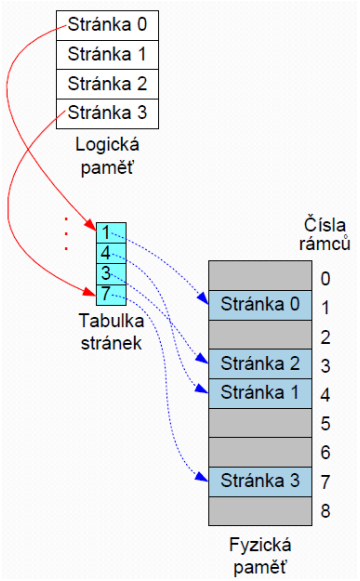
Jednotka správy paměti „stojí“ mezi adresami generovanými programem a skutečnými adresami v operační paměti. Jednotka mění adresy generované programem tak, jak je to momentálně výhodné pro operační systém. Hlavním důvodem pro překlad adres je lepší využití operační paměti.
Druhým důležitým úkolem jednotky správy paměti je zabezpečení ochrany paměti. V moderním operačním systému pracuje zároveň několik programů (mezi nimi i samostatný operační systém). Jednotka správy paměti musí zabránit každému programu v narušení ostatních činnosti ostatních programů nebo samostatného operačního systému. (Nesmí se například stát, že by dva programy používaly stejnou adresu paměti.) [3]
FAP (Fyzický adresný prostor) se dělí na sekce zvané rámce. LAP (Logický adresný prostor) se dělí na sekce zvané stránky. Pevná délka stránek je shodná s délkou rámců. Operační systém udržuje se seznam volných rámců. [16]

Překlad logická adresa →fyzická adresa

- Tabulkou nastavovanou z operačního systému, interpretovanou MMU.
- DAT = *Dynamic Address Translation.*
- PT = *Page Table*, tabulka stránek.
- Řeší problém vnější fragmentace.
- Vnitřní fragmentace (stránky nejsou zcela zaplněny).

Logická adresa (generovaná CPU) se dělí na:

1) Číslo stránky „p“ - index do tabulky stránek, obsahující bázové adresy rámce přiděleného stránce, do které patří logická adresa.
2) Offset ve stránce „*d*“ - relativní adresa (posunutí = *offset*) ve stránce/v rámci.



Obr. 6 *Stránkování* [16]

Implementace tabulky stránek

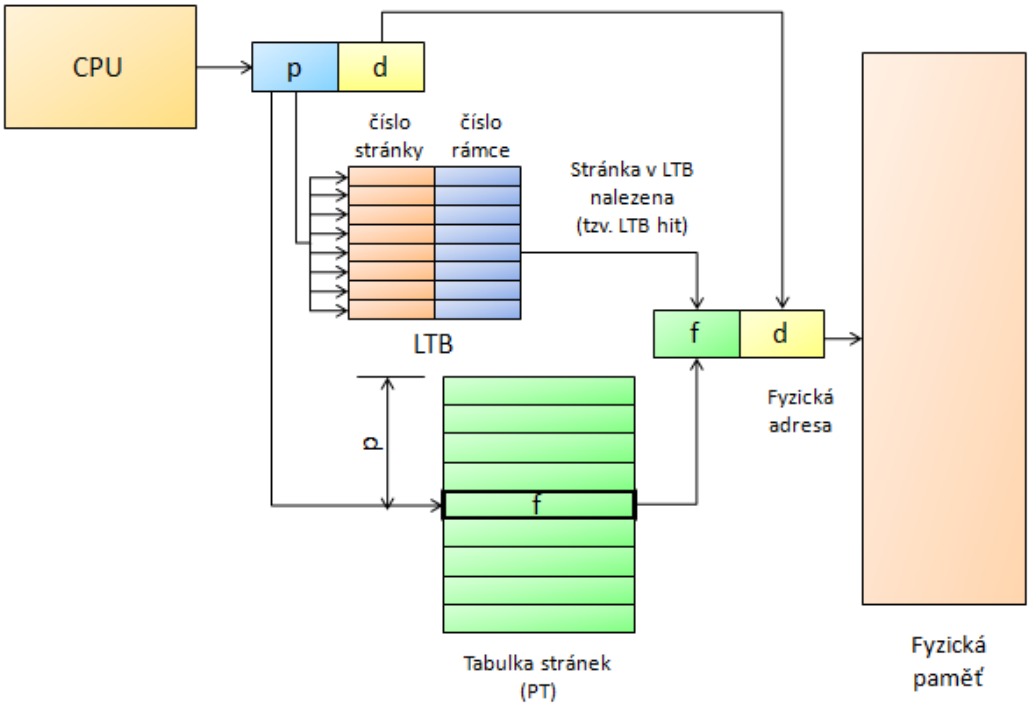
Tabulka stránek je uložena v hlavní paměti. Začátek je odkazován registrem „*Page-table base register*“ (PTBR) - délka je v registru „*Page-table length register*“ (PTLR). Tabulka stránek je víceúrovňová z důvodu urychlení přístupu.

Každý přístup do paměti vyžaduje přístupy dva:

1) přístup do tabulky stránek
2) vlastní přístup do paměti pro údaj/instrukci - časově náročné

Problém se řeší rychlou hardwarovou cache pamětí - *Translation Look-aside Buffer* (TLB):

- Asociativní paměť (paměť „adresovaná“ částí obsahu).
  - Malá kapacita, vysoká rychlost.
  - Jestliže překlad z adresy stránky na adresu rámce není v TLB, získá se z tabulky stránek a provede se aktualizace v TLB. [16]



Obr. 7 *Stránkování s využitím TLB* [16]

Segmentace

Je to podpora uživatelského pohledu na LAP. Program je kolekce (lineárních) segmentů. Každý segment má svůj logický význam – hlavní program, procedura, funkce, objekt a jeho metoda, proměnné, apod.

Dvoudimensionální charakter LAP

- Segment „s“, offset „d“
- Základní úkol implementace – zobrazení do jednodimensionálního FAP

Segmentace se stránkováním

Řeší vnější fragmentaci segmentů pomocí stránkování a je používána na architektuře na x86. Tabulka segmentů – ST obsahuje místo báze segmentu buď na adresu stránkovací tabulky PT, nebo tzv. lineární adresu používanou pro přepočet na fyzickou adresu. [16]

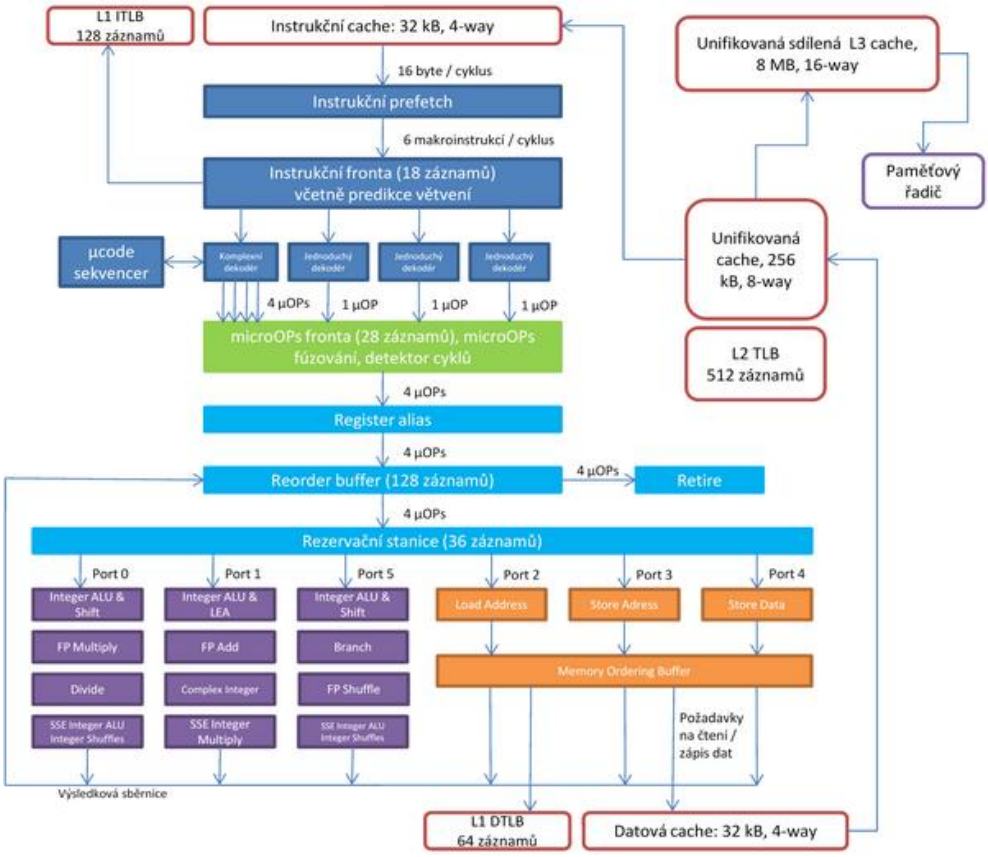
2.1.4 Paměť cache

Je to jakýsi mezisklad dat mezi různě rychlými komponentami počítače. Jeho účelem je vzájemně přizpůsobení rychlostí – rychlejší komponenta čte data z cache a nemusí čekat na komponentu pomalejší (z které si paměť cache data nečetla dopředu).

Do všech mikroprocesorů jsou integrovány malé paměti cache první úrovně, označované jako *First Level Cache* nebo zkráceně L1. Na *[Obr. 8]* je znázorněna cache, která se dělí na část instrukční 32 kB a na část datovou 32 kB.

Slouží k zásobování jednotek mikroprocesoru daty ze sběrnice. Funguje to tak, že do cache se načte ze sběrnice více dat, která pak ve vyrovnávací paměti čekají. Jakmile je mikroprocesor potřebuje, z cache si je načte. Protože cache pracuje rychleji než sběrnice, nemusí mikroprocesor čekat, jak by tomu bylo v případě odebírání dat přímo z pomalejší sběrnice.

Pro zrychlení přesunů dat mezi mikroprocesorem a operační pamětí mají všechny mikroprocesory integrovánu paměť cache druhé úrovně L2. Nové typy procesorů ve svém jádře mají integrovánu další cache paměť – třetí úrovně, označovanou cache L3.



Obr. 8 *Schéma Architektury Nehalem od Intelu*

2.2 Další základní pojmy

2.2.1 Instrukční sada

Musí obsahovat instrukce pro přesuny dat mezi pamětí a registry, aritmetické a logické instrukce, instrukce pro řízení programu a několik systémových instrukcí. Součástí instrukční sady nových mikroprocesorů jsou i instrukce pro koordinaci ve víceprocesorovém prostředí. Výrobci doplňují mikroprocesory o instrukce určené pro přehrávání videa, generování zvuku a grafiky (MMX, 3DNow!, SSE, …).
Pro všechny speciální instrukční sady platí stejné pravidlo:
Aby byly plně využity, musejí je umět používat aplikační programy. Instrukční sady jsou zpětně kompatibilní, v podstatě jen rozšiřují předešlou řadu verzi příkazů.

2.2.2 Vnitřní frekvence (takt)

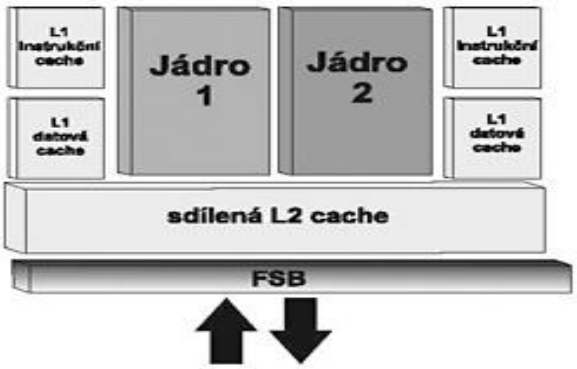
Elektronické obvody fyzicky tvořící mikroprocesor potřebují taktovací impulzy, které určují jejich „pracovní tempo“. Každá základní deska je vybavena generátorem taktů, generujícím taktovací impulzy pro mikroprocesor. Z této externí frekvence je odvozena vnitřní frekvence mikroprocesoru. Mezi externí sběrnicí a mikroprocesorem pracuje tzv. násobička, která převádí pomalejší externí takt na vyšší interní frekvenci mikroprocesoru.
Mikroprocesory mají poměr obou frekvencí pevně určen a měnit je nelze. Násobička (definující poměr vnitřní a vnější frekvence) je součástí mikroprocesoru, proto pokud vsadíme do základní desky nový procesor, nastaví se správný poměr frekvencí automaticky.

2.2.3 Vnější (externí) frekvence

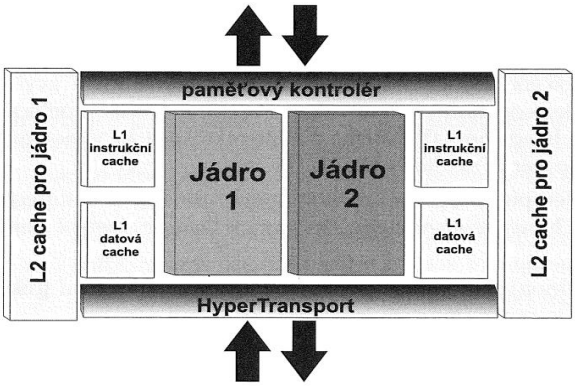
Jde o frekvenci generovanou základní deskou, v jejímž rytmu pracují všechny součástky na desce. Mikroprocesor se z ní prostřednictvím násobičky „vyrobí“ svůj vlastní kmitočet.

2.2.4 Vícejádrový procesor

Pro zvýšení výkonu se do mikroprocesoru umisťují více jader. Mikroprocesor pracuje ve stejné patici jako jeho předchůdci, ale uvnitř procesoru je integrováno více jader (nyní 2 nebo 4).



Obr. 9 *Dvoujádrový procesor Intel* [3]



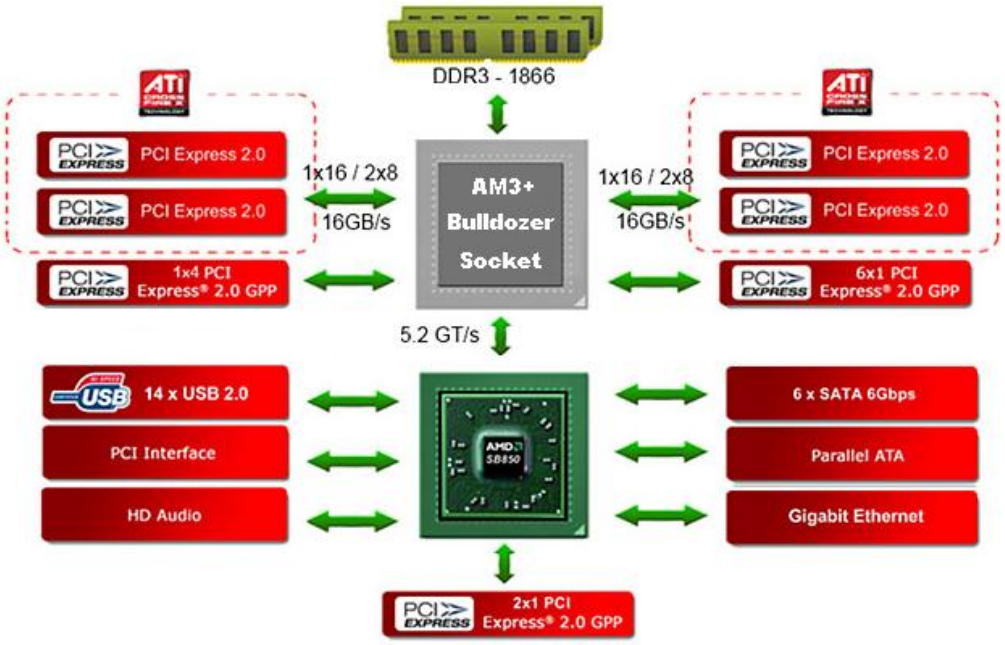
Obr. 10 *Dvoujádrový procesor AMD* [3]

2.2.5 Hyper-Threading

Umožňuje procesoru „tvářit se“, jako by v počítači byly procesory dva – jeden fyzický procesor se chová jako dva logické. Jde o to, že jednotlivé části procesoru nejsou v průběhu vykonávání výpočtů (běhu programu) využity stoprocentně. Při provádění jednoho výpočetního vlákna (anglicky thread) se vyskytují okamžiky, kdy do určité části jádra procesoru zahálí. Tehdy se nevyužitá jednotka zapojí do výpočtu druhého programovatelného vlákna, a tak se docílí zvýšení výkonu, protože se vlastně dvě věci počítají souběžně. Stále však jde o jeden procesor, který určité systémové prostředky (např. cache paměť atd.) sdílí pro obě vlákna, a proto není nárůst výkonu dvojnásobný, ale maximálně 30%. [3]

2.2.6 Hypertransport

Je to vysoce výkonná páteřní sběrnice osobních počítačů. Principem Hypertransportu je paketový systém přenosu, který formou tunelů umožňuje snadno implementovat příslušnou externí sběrnici do systému s HyperTransport technologií. [4]



Obr. 11 *Uspořádání mcchipsetu se sběrnicí HyperTransport - AMDBulldozer*

3 TECHNOLOGIE INTEL VIRTUALIZATION TECHNOLOGY - IVT

3.1 Technologie VT-x

Hardwarová podpora virtualizace na procesoru x86 byla poprvé představena v roce 2005 pod označením VT-x. [19]

Popis práce technologie Intel VT-x

Uživatelská aplikace běží v úrovni ochrany 3 a jádro operačního systému virtuálního počítače na úrovni 0. Prostředí běhu se neliší od nativního, takže není třeba směrovat systémová volání ani upravovat jádro. Každé vyvolání privilegované instrukce hostovaným OS nebo způsobení výjimky vyvolá hardwarově implementovanou operaci VM-exit. Řízení převezme VMM, který bezpečně provede privilegovanou operaci nebo obslouží výjimku a vyvolá také hardwarově implementovanou operaci VM-entry, čímž se provede přechod zpět k virtuálnímu stroji. [1, 19]

Technologie VT-x představuje dvě režie činnosti procesoru:

- VMXON operace představuje úroveň ochrany *ring -1*. Zde běží VMM, který má možnost provádět nejvíce privilegované operace.

- VMXOFF operace rozumíme úroveň, na které běží virtuální stroj. K realizaci přechodu mezi režimy se využívají hardwarově implementované operace označované VM-entry a VM-exit.

Obě tyto formy režie jsou podporovány na všech čtyřech úrovních oprávnění.

Popis operace VM-entry: [1, 18]

- instrukce VMLAUNCH (stav běžící) a VMRESUME (pozastaveno)
- přechod z VMM =>VM
- vstoupí do režimu, ve kterém běží virtuální stoje VMXOFF
- načte stav VM z VMCS

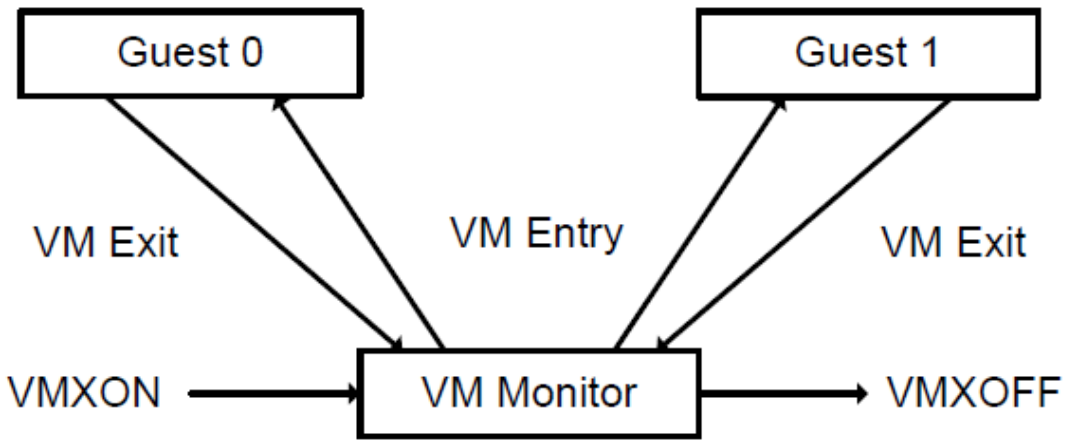
Popis operace VM-exit: [1, 18]

- instrukce VM-exit
- přechod VM => VMM
- vstoupí do VMXON
- uloží stav VM do VMCS
- načte VMM stav z VMCS
- přechod může být vyvolán z více důvodů, např. při vyvolání výjimky nebo vyvolaní privilegované instrukce hostovaným OS

3.1.1 Životní cyklus VM software

Sumarizace životního cyklu:

- SW vstoupí jako VMX operace spouští se v procesoru jako VMXON instrukce.
- Použití VM záznamů, VMM zadává procesy jeden po druhém, které mohou nabývat stavu VMLAUNCH (běžící), VMRESUME (pozastaveno). Tím se získá kontrola na případné ukončení procesů.
- VM může přijmout opatření k ukončení procesu i tak, že proces stane na vstup VM.
- Nakonec se může rozhodnout, zda se VMM vypne a nechá se běžet VMX provoz. Dělá se to užitím VMXOFF instrukce. [5]



Obr. 12 *Životní cyklus architektury Vt-x* [5]

3.1.2 Kontrolní struktura virtuálního počítače

Každý přechod mezi VM a VMM (VM-exit / VM-entry) vyžaduje velké množství cyklů CPU. Počet těchto režijních cyklů závisí na interní architektuře CPU a prováděné operaci. Provádění takových operací může zabrat více než několik stovek až tisíce cyklů.

I když jsou operace přechodu mezi VM a VMM implementované hardwarově, jsou to operace, které výrazně zatěžují procesor.

Ztráta způsobená přechodem mezi VM a VMM má menší dopad na provádění složitých operací. Za složité operace lze považovat systémová volání, protože ty vždy spotřebují velké množství cyklů CPU. Jednoduché operace, jako je vytváření procesu, změna kontextu, malé změny v tabulce stránek, spotřebují v nativním prostředí jen velmi málo cyklů. Přechod VM / VMM v tomto případě znamená mnohonásobné navýšení počtu spotřebovaných cyklu CPU.

VMXOFF a přechody jsou řízeny datovou strukturou (zásobníkem), tzv. řídící strukturou VMCS *Virtual Machine Control Structure* [5]. VMCS jedná se o skupinu tabulek, které jsou uloženy v paměti cache.

Přístup k VMCS je řízen prostřednictvím součásti procesoru VMCS (jeden na logický procesor) – volání ukazatele. Do/z ukazatele VMCS lze číst a zapisovat pomocí pokynů VMPTRST a VMPTRLD. VMM konfiguruje pomocí VMCS VMREAD, VMWRITE a VMCLEAR pokynů.

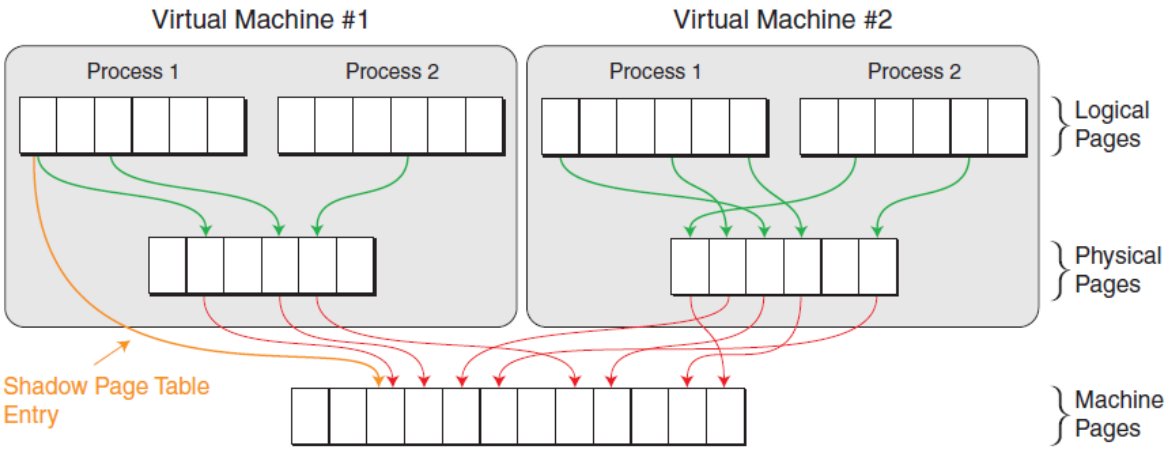
VMM může používat jiný VMCS pro každý virtuální stroj, který ji podporuje. Pro virtuální stroj s více logickými procesory (virtuální procesory), by mohl VMM použít různé VMCS pro každý virtuální procesor. [1, 5]

3.1.3 SW a HW podpora virtualizace stránkování

a)Softwarová podpora stránkování

Jsou-li virtuální počítače provozovány na VMM *Virtual Machine Monitor* [1], nebudou mít hostované operační systémy přístup k tabulkám hardware stránku jako nativně spouštět operační systémy. VMM emuluje tabulky stránek pro hosta (virtuální operační systémy) a dává hostující operační systémy iluzi, že je skutečný fyzický přístup čísla stránek při mapování z logických čísel a spuštěné procesy. VMM využívá vlastní tabulky stránek nazývané stínové tabulky stránek *Shadow Page Tables* [6], z nichž je vidět na hardware

systému. Takže vždy, když hostující OS požádá o virtuální adresu překlad do fyzické paměti, vyřízení žádosti je zachyceno. VMM podle pořadí začne procházet své stínové tabulky a poskytne adresu fyzické paměti. [6]



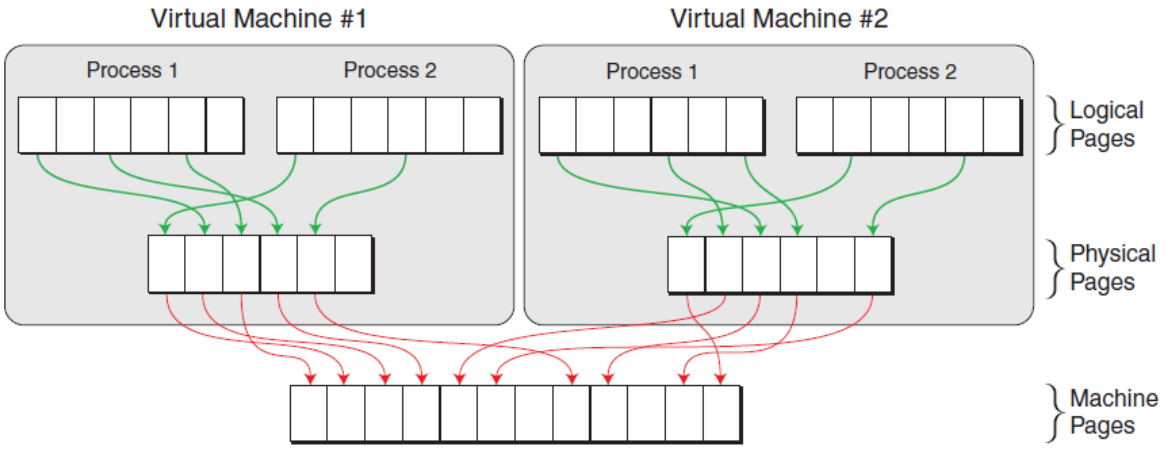
Obr. 13 *Softwarová podpora stránkování* [7]

b)Hardwarová podpora stránkování – EPT

EPT *Extended Page Tables* [7] se stará o paměť, režijní náklady a poskytuje virtuálnímu stroji provádět vše mnohem rychleji než software nebo VMM (přístup do paměti). Hardwarová podpora pro správu paměti pomocí EPT usnadňuje provoz VMM. Jednotlivé virtuální stroje jsou sledovány pomocí TLB, která jim přidělí identifikátor adresního prostoru. Pomocí adresového prostoru identifikátor TLB může sledovat virtuální stroj adresového prostoru a nemusí vypráznit TLB cache, jestliže jeden VM přepíná prostor.

Používáním EPT operační systém hosta pokračuje ve spravování LPN->PPN mapování tabulek stránek hosta, ale VMM udržuje PPN->MPN *Machine Pages* [7] mapování navíc v další úrovni tabulky stránek, nazývané vnořené tabulky stránky *Nested Page Tables* [7]. V tomto případě jsou obě tabulky hosta a vnořené tabulky stránky přístupné hardwaru. Když je přistupováno na logickou adresu, hardware prochází tabulkou stránky hosta jako v případě nativního spuštění, ale pro každou PPN přistupuje během procházení tabulky stránky hosta. Hardware také prochází vnořené tabulky stránky k tomu, aby určil odpovídající MPN. Tento složený překlad odstraňuje nutnost udržovat shadow page tables a synchronizuje je s tabulkami stránek hosta. Nicméně přidaná operace také zvyšuje

náročnost procházení stránek, proto má dopad na výkon aplikací, které zatěžují TLB. Tato náročnost může být snížena jak použitím velkých stránek, tak snižováním zatížení na TLB pro aplikace s vhodným umístěním. [6, 7]



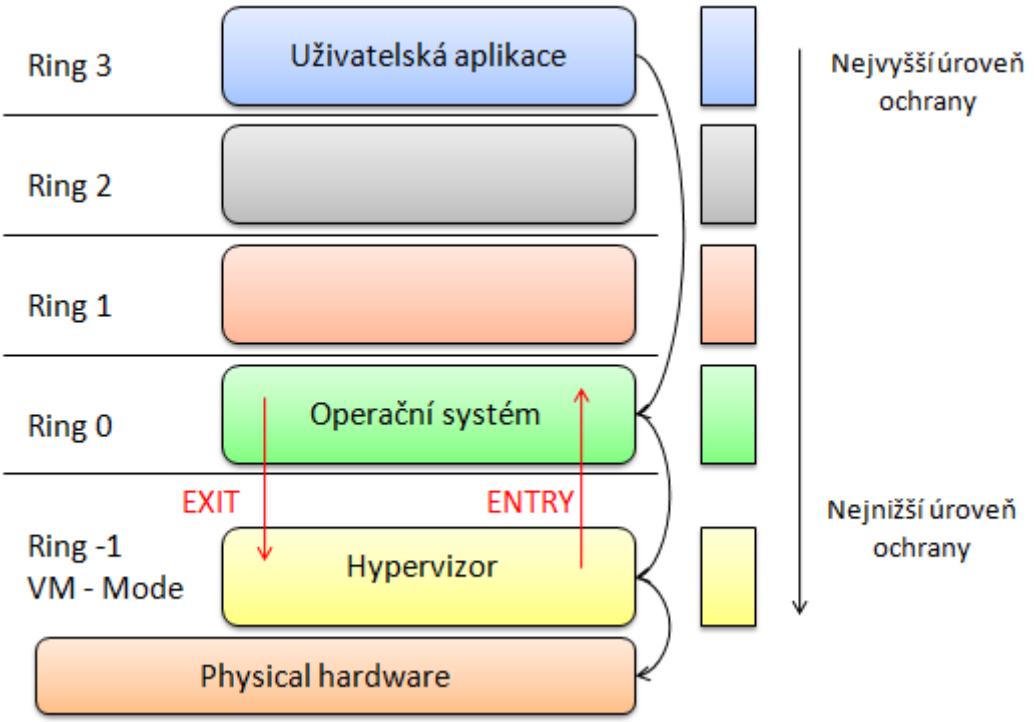
Obr. 14 *Správa paměti pomocí EPT* [7]

4 TECHNOLOGIE AMD VIRTUALIZATION – AMD-V

4.1 Technologie SVM

Technologie SVM *Secure Virtual Machine* byla založena v roce 2006, tzn. o rok později než technologie IVT *Intel Virtualization Technology*. Velkou výhodou této technologie je to, že začala vyrábět CPU podporující hardwarovou virtualizaci hned na x64 bitové architektury. [8]

Technologie SVM je podobná technologii VT-x.



Obr. 15 *Architektura SVM na 4 úrovních oprávnění*

Technologie SVM je založena na instrukci VMRUN *Virtual Machine Run* [8]. VMRUN instrukce provedená HW umožní VM spustit v „VM Mode“. VM se pak vrací zpět přes VM-Exit do hypervizoru. Hypervizor pak následně pokračuje na základě instrukce VMRUN.

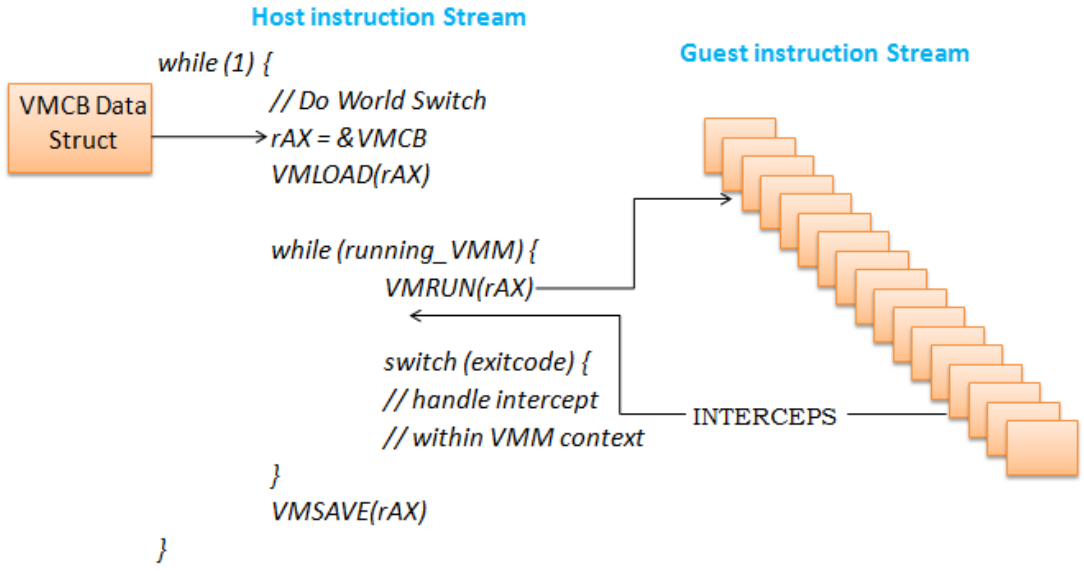
Stejně jako u VT-x tak i technologie SVM využívá dvě operace.

Popis operace VMRUN- Entry: [8]

- VMM je ukládán do paměti
- VM načte stav z VMCB
- Vstupuje do režimu, ve kterém běží virtuální stroje

Popis operace VMRUN- Exit: [8]

- Přechod z VMM do VM
- Ukládá stav VM do VMCB
- Načítá stav VMM z VMCB



Obr. 16 *VMRUN instrukce* [8]

Všechny CPU pro VM jsou umístěny ve Virtual Memory Control Block (VMCB) datové struktury. [8]

4.1.1 VMCB – Virtual Machine Kontrol Block

Podobně jako u Intelu (*viz. kapitola [3.1.2] Kontrolní struktura virtuálního počítače),* tak i u architektury AMD jsou přechody řízeny datovou strukturou (zásobníkem), tzv. řídící

strukturou VMCB *Virtual Machine Control Block* [8]. Pracuje s instrukcemi VMRUN, VMLOAD, VMSAVE, VMMCALL a další. [8]

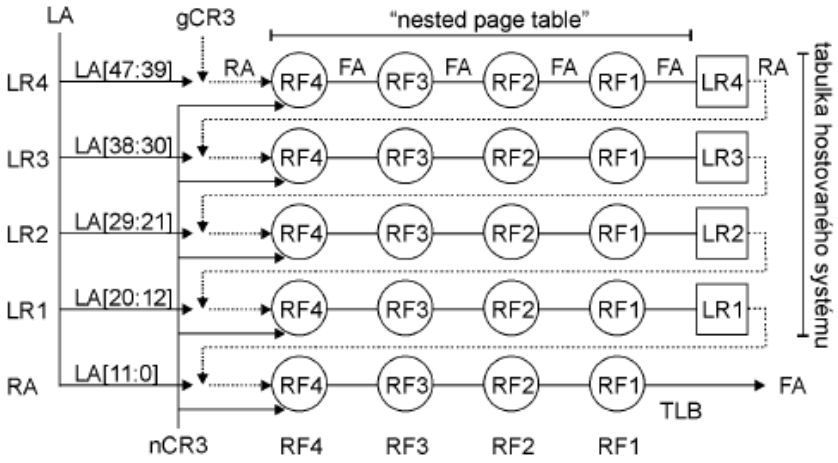
4.1.2 HW podpora virtualizace stránkování NPT

HW podpora virtualizace stránkování je popsána v [9]. NPT *Nested Page Table* je rozšíření MMU, které během překladu adresy zpracovává MMU najednou dvě tabulky stránek. Jednou z nich je tabulka hostovaného systému, která obsahuje překlad virtuálních na reálné adresy. Druhou tabulkou je nově implementovaná tabulka hypervizoru, tzv. Nested Page Table, která obsahuje překlad reálných na fyzické adresy. Výsledek překladu následně uloží MMU do TLB. Na rozdíl od stínových tabulek stránek, používaných při plné virtualizaci, obsahuje TLB celý překlad z lineární adresy hostovaného systému na fyzickou adresu. Stínové tabulky také vyžadovaly pravidelnou synchronizaci s primárními tabulkami stránek hostovaného systému. Tento problém u nested page table odpadá, protože MMU přímo přistupuje k oběma tabulkám.

Průchod MMU jak tabulkou stránek hostovaného systému, tak NPT, ale sám o sobě přidává určitou režii. Pro čtyřúrovňové stránkování se může překlad prodloužit ze čtyř přístupů k paměti na 24 přístupů. Každou ze čtyř bází tabulek stránek musí MMU přeložit na fyzickou adresu pomocí NPT a následně přeložit bázi cílové stránky. Částečného zlepšení dosáhneme změnou velikosti stránek. Velké stránky snižují počet úrovní potřebných pro překlad adresy. Můžeme zvětšit jak velikost stránek v NPT, tak stránky v tabulce stránek hostovaného systému. Zvětšením stránek také nepřímo zvýšíme efektivitu TLB, protože stejný počet záznamů v TLB nyní adresuje větší množství paměti.

Efektivita nested paging významně závisí na dobrém naplnění TLB. Obecně se snažíme v TLB udržet co nejvíce záznamů po co nejdelší dobu, aby nedocházelo k častým TLB miss, vedoucím k drahým překladům. Jedna z technik zavedených společně s nested paging zabraňuje vyprazdňování TLB při přepínání kontextů. Hypervizor přiděluje jednotlivým hostovaným systémům jedinečný identifikátor, takzvaný ASID (Address Space ID), který se ukládá ke každému záznamu v TLB. Každý systém má přístup pouze ke třem záznamům, které jsou označeny jeho identifikátorem. Docílíme tak dynamického rozdělení TLB mezi více hostů a samotný hypervizor, přičemž obsah TLB nemusíme kvůli přepnutí mezi hostovanými systémy vyprazdňovat. Obecně tuto techniku nazýváme tagged TLB. Další možností, jak zvýšit efektivitu překladu, je zavést speciální TLB vyhrazenou pro

ukládání překladů z reálné adresy na fyzickou adresu. Redukujeme tak počet přístupů do paměti během překládání bázových adres tabulek stránek. [9]



LA: logická adresa
RA: reálná adresa
FA: fyzická adresa

LRk: k-tá úroveň překladu z logické na reálnou adresu
RFk: k-tá úroveň překladu z reálné na fyzickou adresu
gCR3: registr CR3 hostovaného systému
nCR3: registr CR3 fyzického systému

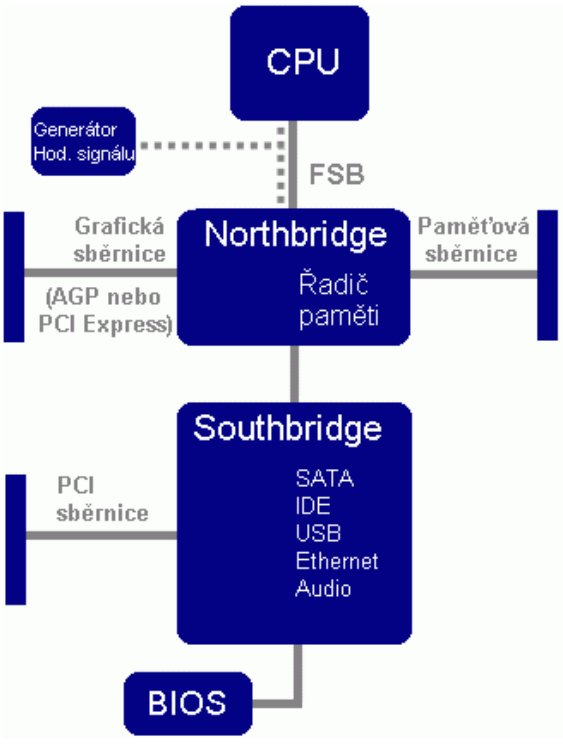
Obr. 17 *Nested paging* [9]

5 INTEL vs AMD

5.1 Procesory Intel

Jednotlivá jádra od Intelu sdílela jednotnou sběrnici FSB. Paměťový řadič byl umístěn na severním můstku, takže když chtěl procesor přistupovat k RAM paměti, musel jít přes severní můstek. Neefektivní je rovněž použití Untagged TLB *Translation Lookaside Buffer*. To je pomocná tabulka, kde si procesor pamatuje několik posledně použitých virtuálních adres. V praxi velmi často sahá na stále stejná místa v paměti, pokud si na tato místa pamatuje přesnou cestu, kterou nemusí vždy složitě překládat a zjišťovat, mnohem rychleji se k požadovaným datům dostane. Problémem Untagged TLB je, že při přepnutí jiného virtuálního stroje se musí TLB vyprázdnit.

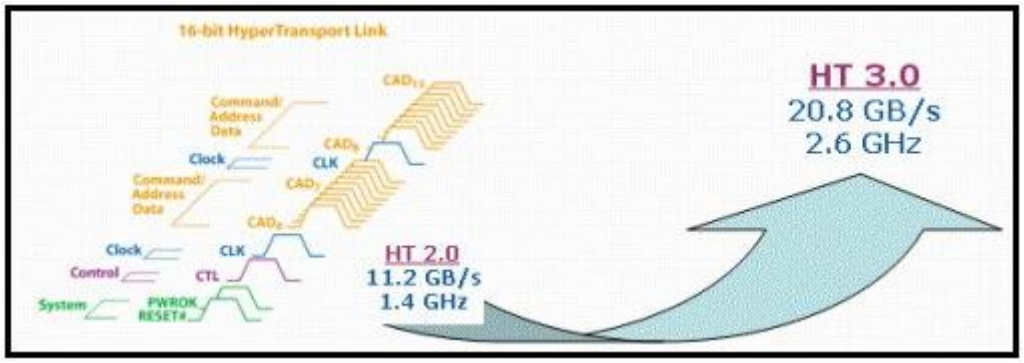
Dalším problémem architektury Intel je to, že operační paměť není virtualizovaná, management paměti tak musí být řešen softwarově. [10]



Obr. 18 *Přistupování CPU k RAM paměti přes severní můstek*

5.2 Procesory AMD

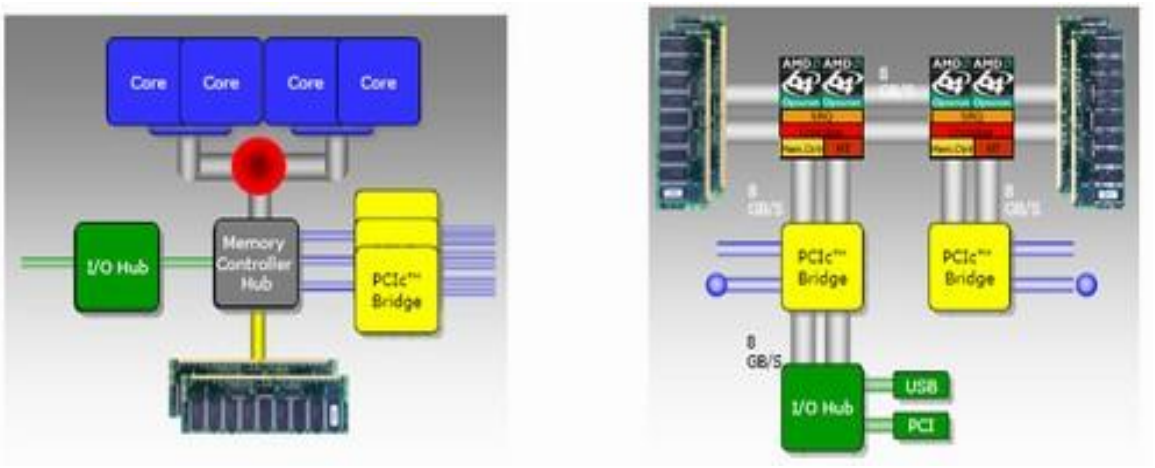
AMD využívá Hyper-transport 3.0, která je mnohem rychlejší a efektivnější než zatížená jednotná FSB sběrnice. Maximální takt sběrnice je v obou směrech 5,2 GHz a propustnost 20,8 GB/s v jednom směru.



Obr. 19 *Hyper-transport 3.0 u AMD* [12]

Zároveň využívá Tagged TLB [1] 1 (je zde přidán identifikátor adresního prostoru ASID), při přepínání se pak TLB vyprazdňovat nemusí, což šetří paměťové operace.
AMD má integrovaný paměťový řadič operační paměti přímo na procesoru, čímž se stává méně závislým na čipsetu a zároveň tak efektivně izoluje jednotlivé virtuální stroje v paměti hardwarovou cestou. [10, 12]

[1] Pozn.:[1] Tagged Translation Lookaside Buffer (TLB), je tabulka, kde si procesor pamatuje několik posledně použitých virtuálních adres. Při přepínání mezi systémy se Tagged TLB nemusí vyprázdnit, což vede k dalšímu zrychlení.



Obr. 20 *Čtyř-jádrové achitektury Intel (vlevo) a AMD (vpravo)* [10]

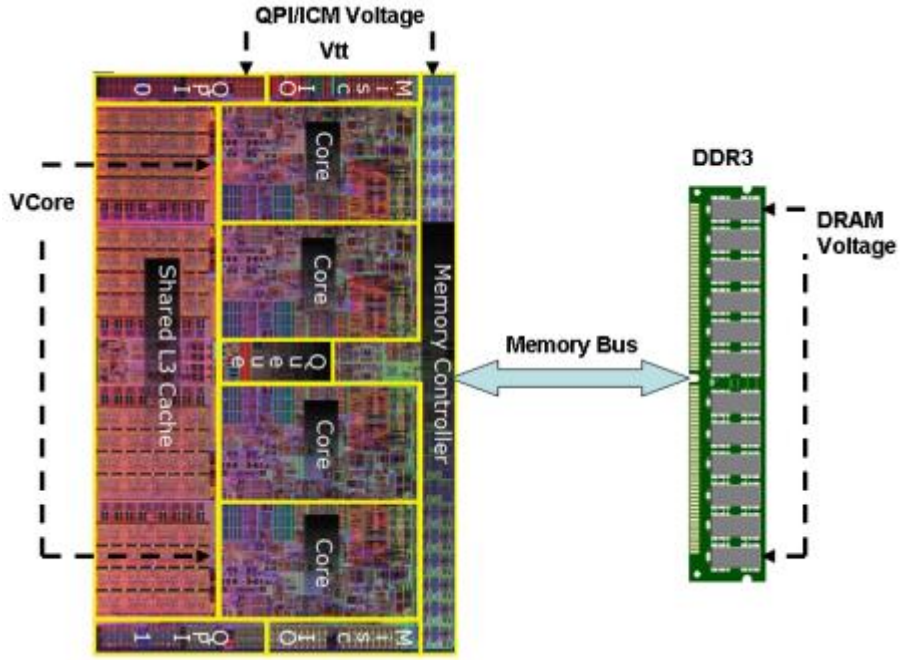
5.3 Architektura Nehalem od Intelu

5.3.1 Core i7

V roce 2008 Intel přišla s novou verzi nových procesorů z řady Core pod označením i7. Intel nahradila stávající datovou sběrnici FSB *Front Side Bus* a přemístila řadič operační paměti přímo k procesoru.

Integrovaný řadič paměti umožňuje zlepšení výkonu, dokáže konsolidovat více virtuálních strojů na méně fyzických systémů a umožní virtualizovat aplikace, které nemohly být dříve virtualizované. [11]

Byla stvořena sběrnice QPI *Intel QuickPath Interconnect*. Lze se setkat také s dřívějším formálním označením Common System Interface - CSI. Jedná se o point-to-point systém přenosu založený na paketové komunikaci. Díky vzájemnému propojení jednotlivých komponent namísto jejich poměrně zdlouhavého připojení k původní FSB je ušetřen čas během putování informací ke svému cíli. [13]



Obr. 21 *Nehalem i7 integrovaný paměťový řadič*

6 PŘÍKLADY POUŽITÍ U VYBRANÝCH TECHNOLOGIÍ VIRTUALIZACE V PRAXI

V úvodu této práce byly představeny 3 techniky virtualizace s podporou hypervizoru.

1. Úplná virtualizace bez podpory procesoru – nejvyužívanější technikou je Binární překlad – nejznámější aplikace, které využívají úplné virtualizace, jsou např. VMware, VirtualBox, Xen

2. Paravirtualizace – nejznámějším softwarem s možnosti paravirtualizace je Xen.

3. Hardwarová virtualizace – software, které podporují tuto techniku patří VMware, Hyper-V, Xen.

Společnost VMware byla první, která svůj hypervizor zpřístupnila přímo v hardwaru serveru. ESXi společnosti VMware je malý 32MB hypervizor.

ESXi je bezplatný integrovaný hypervizor, dostupný s hardwarem nebo je samostatně dostupný ke stažení. Běží na virtuálních počítačích platformy x86 a x64. Vyžaduje přidání jedné z verzí tohoto produktu, aby bylo možné přitoupit ke všem funkcím hypervizoru.

Další společnosti, které dodávají konfiguraci hypervizorů integrovaných s hardwarem je společnost Citrix, která nabízí verzi OEM svého XenServeru a také společnost Microsoft nabízí integrovanou verzi Hyper-V. [14, 17]

II. PRAKTICKÁ ČÁST

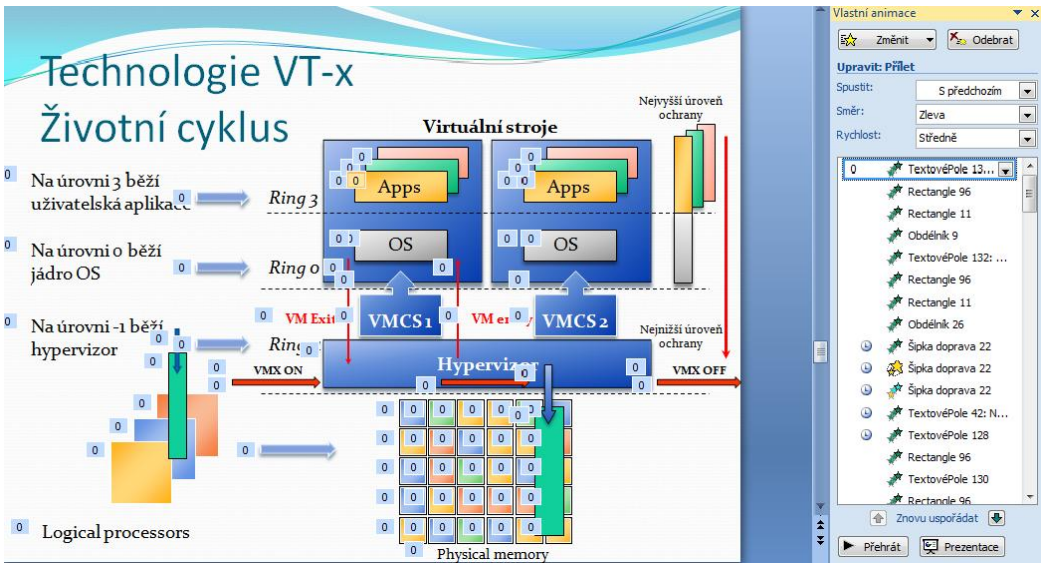
7 VIZUALIZACE HARDWAROVÝCH PRINCIPŮ VIRTUALIZACE

Hardwarové principy virtualizace byly vizualizovány v aplikaci Microsoft Office PowerPoint. Pro tvorbu této prezentace bylo vycházeno z kapitol, které jsou uvedeny v teoretické části této bakalářské práce. K tvorbě prezentace jsem si vybrala technologii VT-x od IVT, která je totožná s technologií SVM od AMD-V.

Prezentace obsahuje:

- Obecný popis hardwarové virtualizace,
- základní pojmy virtualizace – virtuální stroj, ring, hypervizor,
- VMXON a VMXOFF operace,
- popis operací VM-entry a VM-exit,
- životní cyklus technologie VT-x,
- VCMS,
- stránkování.

Jednotlivé prezentace obsahují krátké věty a obrázky, pro snadné pochopení tohoto tématu. Obrázky byly animovány přes záložku animace => vlastní animace, kde byly jednotlivé funkce načasovány tak, aby z nich byla patrná návaznost při probíhající virtualizaci.



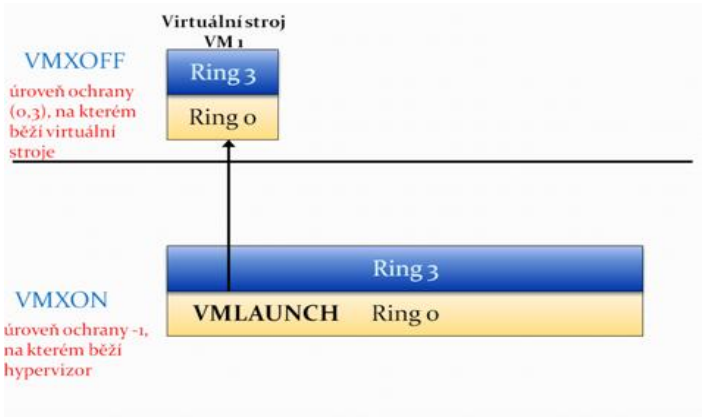
Obr. 22 *Hlavní okno prezentace*

- Na *[Obr. 22]* je zároveň znázorněn životní cyklus hardwarové virtualizace
- Virtualizace běží nad HW.
- Je rozšířena o další úroveň oprávnění ring -1, na které běží hypervizor.
- Na úrovni ring 0 běží OS např. Linux, Microsoft XP, apod.
- Úroveň ring 3 je určen pro běh uživatelské aplikace.

* *Více je uvedeno v kapitolách [1.4] Virtualizace s podporou procesoru, [3.1] Technologie VT-x*

7.1 Popis operace VMXON a VMXOFF – instrukce VMLAUNCH (běžící) a instrukce RESUME (pozastaveno)

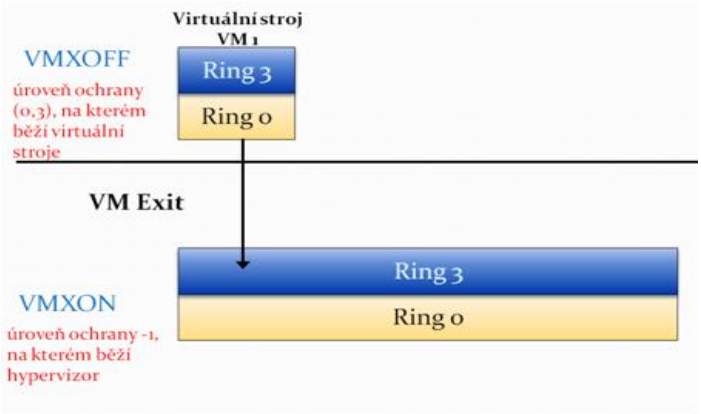
Krok 1



Obr. 23 *Popis operace VMXON a VMXOFF – krok 1*

- V ringu 0 se spustí stav VMXON. Poté se spouští operace VM-Entry pomocí instrukce VMLAUNCH (běžící).

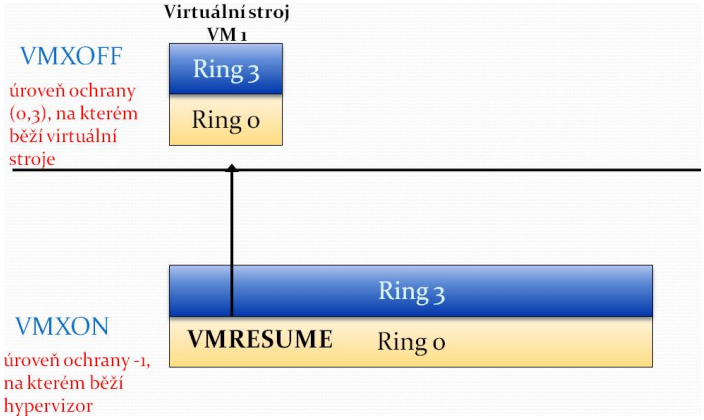
Krok 2



Obr. 24 *Popis operace VMXON a VMXOFF – krok 2*

- Po operaci VM-Entry se spustí stav VMXOFF pomocí instrukce VM-Exit.

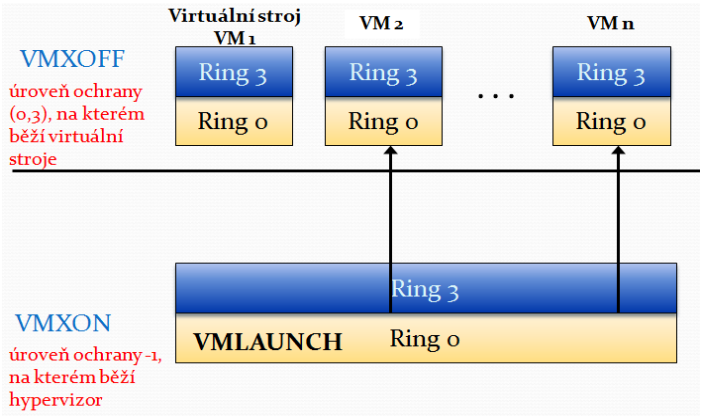
Krok 3



Obr. 25 *Popis operace VMXON a VMXOFF – krok 3*

- Ze stavu VMXON pomocí operace VM-Entry s instrukcí VMRESUME (pozastaveno) se poté budou spouštět další virtuální stroje.

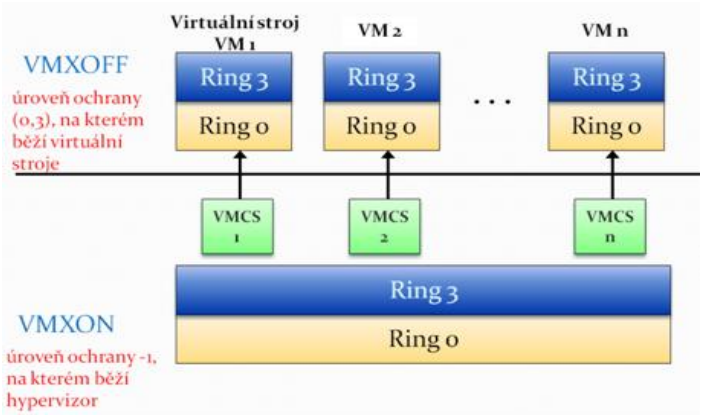
Krok 4



Obr. 26 *Popis operace VMXON a VMXOFF – krok 4*

- Poté spustí další virtuální stroje opět pomocí operace VM-Entry pomocí instrukce VMLAUNCH.

Krok 5



Obr. 27 *Popis operace VMXON a VMXOFF – krok 5*

- Pomocí operace VM-Exit se ukládá stav VM do tabulek VMCS, která je uložena v paměti cache a vstupuje do stavu VMXON.
- Po opětovném spuštění se uložená data z VM načítá z tabulek VMCS pomocí operace VM-Entry a načte stav VMM z VMCS.

ZÁVĚR

Cílem této práce bylo představit virtualizaci s podporou procesoru. V úvodu práce byl představen pojem virtualizace, techniky virtualizace s podporou hypervizoru a možnosti jejího využití v praxi. Dále jsem uvedla základní principy technologie VT-x od Intelu a technologii SVM od AMD. Hlavním úkolem této práce bylo porovnat obě uvedené technologie.

Společnost Intel uvedla technologii VT-x v roce 2005 a rok později v roce 2006 AMD uvedla technologii SVM.

AMD začala vyrábět procesory s podporou virtualizace hned na x64 architektuře, což bylo mnohem efektivnější, protože 64bitové procesory dokážou adresovat více paměti než architektury x86. Dále se procesory od AMD lišily integrovaným řadičem přímo do procesoru, což umožňovalo, že se stal méně závislým na čipsetu a zároveň tak efektivně izoloval jednotlivé virtuální stroje v paměti hardwarovou cestou.

Intel měl integrovaný řadič v čipsetu na severním můstku, takže když chtěl procesor přistupovat k RAM paměti, musel jít přes severní můstek. Dalším problémem architektury Intel bylo to, že operační paměť nebyla virtualizovaná, management paměti tak musel být řešen softwarově. Teprve až v roce 2008 Intel přišla s novou verzí nových procesorů z řady Core pod označením i (i3, i5, i7), která měla integrovaný řadič na procesoru.

Dále jsem v této práci představila hardwarovou podporu paměti. První generace obou technik neobsahovala podporu virtualizace paměti. AMD uvedla svou techniku v roce 2007 pod označením NPT *Nested Page Tables* a Intel až o dva roky později v roce 2009 s technikou EPT *Extended Page Tables*. Obě tyto techniky snižují vyšší režii a zároveň umožňují hostovanému systému spravovat vlastní tabulky stránek a také ošetřovat výpadky stránek.

V praktické části jsem vizualizovala technologii VT-x od Intelu v programu PowerPoint. Jednotlivé prezentace obsahují obrázky a krátké popisné texty, pro snadné pochopení tohoto tématu. Jednotlivé obrázky jsou animovány tak, aby z nich byla patrná časová návaznost při probíhající virtualizaci operačních systémů.

Procesory od Intelu a AMD jsou v dnešní době koncepčně podobné.

ZÁVĚR V ANGLIČTINĚ

The objective of this work was introducing the virtualization with the processor support. In the introduction has been presented the conception of virtualization, the techniques of virtualization with the support of hypevisor and some possibilities its employ in a practice. I introduced the elementary principles of technology VT-x from Intel and the technology SVM from AMD. The primarily intentions this work was compare the both introduced technology.

The Intel company introduced VT-x technology in 2005 and the AMD installed SVM a year later.

The AMD installed the processors with the support of the virtualization right on to x64 architecture what was more effective because 64-bit processors can address more memory than the x86 architecture. The AMD processors are different with the integrated controller direct into the processor what was enable to become more independent of chipset and at the same time isolated the individual virtual machine by hardware way in the memory.

The Intel had integrated the controller in the chipset in the northbridge so when the processor wanted to access RAM memory it had to go over the north bridge. There was other problem of the architecture Intel the operating memory wasn't virtualizated and the management of the memory had to tried to solve as a software. The Intel came with the new version of the processors from the Core i (i3, i5, i7) serie that had integrated the controller in the processor.

Furthermore in my work i introduced the hardware support of memory. The first generation the both techniques didn't contain any support of the memory virtualization. The AMD intruduced their technique in 2007 and was marked NPT *Nested Page Tables* and The Intel installed their technique EPT *Extended Page Tables* two years later in 2009. The both these techniques reduce the high-ranking overhead and at the same time let to guest system take care for one's own table of the pages and service some outage pages.

I vizualizated the technology VT-x by The Intel in the program PowerPoint in the practical part. The individual attendances contain the pictures and the short descriptive texts to better understanding this topic. The picture are animated to be known the time sequence at ongoing virtualization of the operating systems.

Presently the processors from the Intel and from the AMD are conceptually similar.

SEZNAM POUŽITÉ LITERATURY

[1] KRÁL, Jan; KRAHULEC, Lukáš. *WikiHosting* [online]. 5.12.2008 [cit. 2011-01-31]. PAP_Virtualizace_referát_(Krahulec_Král).pdf. Dostupné z WWW: <http://wh.cs.vsb.cz/mil051/images/f/f5/PAP_Virtualizace_refer%C3%A1t_%28Krahulec_Kr%C3%A1l%29.pdf>.

[2] BERAN, Radek. *Mikroprocesor* [online]. 2003-05, 2010-03-15 [cit. 2011-01-31]. Radek Beran's Homepage. Dostupné z WWW: http://www.beranr.webzdarma.cz/hardware/mikroprocesor.html.

[3] HORÁK, Jaroslav. *Hardware: učebnice pro pokročilé*. 4. aktualizované vydání. Brno : Computer Press, a.s., 2007 . 360 s. ISBN 978-80-251-1741-5.

[4] SYSEL, Martin. *Technické vybavení PC*. Vyd.1. Zlín: Univerzita Tomáše Bati, Fakulta technologická, 2003. 138 s. ISBN 80-7318-108-8.

[5] *Intel* [online]. Intel Corporation: Copyright, 2007-2010 [cit. 2010-10-31]. Dostupné z WWW: <http://www.intel.com/Assets/PDF/manual/253669.pdf>.

[6] GOWDA, Bhaskar . *Intel* [online]. 3.6.2009 [cit. 2011-01-31]. The Server Room Blog. Dostupné z WWW: <http://communities.intel.com/community/openportit/server/blog/2009/06/02/a-peek-into-extended-page-tables?wapkw=%28Extended+page+tables%29%29>.

[7] *VMware* [online]. 2009-03-30 [cit. 2011-03-27]. VM_ESX_Intel-EPT-eval.pdf. Dostupné z WWW: <http://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf>.

[8] O'BRIEN, David. *AMD* [online]. 2006-05-10 [cit. 2011-03-29]. 9-David_Obrien-AMD_India_SVM_DO_v1.pdf. Dostupné z WWW: <http://developer.amd.com/Assets/9-David_Obrien-AMD_India_SVM_DO_v1.pdf>.

[9] Patka, L. *Virtualizační technologie.* Brno, 2009. Diplomová práce na Fakultě informatiky Masarykově univerzity v Brně. Vedoucí práce diplomové práce Ing. Mgr. Zděněk Říha, Ph.D.

[10] ŠURKALA, Milan. AMD představuje novou generaci Opteronů - Socket F. AMD představuje novou generaci Opteronů - Socket F [online]. 2006 [cit. 2011-01-31]. Dostupný z WWW: http://www.svethardware.cz/art_doc-DA393B9268E69CEEC12571C70080750F.html.

[11] *EWeek.com* [online]. 2009-03-30 [cit. 2011-03-29]. Intel Nehalem Will Give Virtualization a Boost - Virtualization - News & Reviews eWeek.com. Dostupné z WWW: http://www.eweek.com/c/a/Virtualization/Intel-Nehalem-Will-Give-Virtualization-a-Boost-324344/.

[12] OBERMAIER, Z. *PC Tuning* [online]. 14.5.2007 [cit. 2011-04-02]. *AMD nové produkty - nová éra začíná*. Dostupné z WWW: <http://pctuning.tyden.cz/multimedia/zpravy-it/8797-amd_nove_produkty-nova_era_zacina>.

[13] LALÍK, Aleš . *Intel QuickPath - rychlejší cesta k datům* [online]. 2008-08-06 [cit. 2011-04-03]. Notebook.cz. Dostupné z WWW: <http://notebook.cz/clanky/technologie/2008/Intel-QuickPath>.

[14] *Manag* [online]. 9-2008 [cit. 2011-05-31]. Dostupné z WWW: http://www.manag.com/public/data/data/Virtualizace_v1.3.pdf

[15] GOLDEN, Bernard. *Virtualization for Dummies* [online]. Indianapolis, Indiana : Wiley Publishing, Inc., 2009 [cit. 2011-05-31]. Dostupné z WWW: <http://media.wiley.com/assets/2109/45/9780470607527.pdf>. ISBN 978-0-470-52675-0.

[16] Sysel, M.: Operační systémy - GNU/Linux, Skripta UTB ve Zlíně, 2006

[17] RUEST, Danielle ; RUEST, Nelson . *Virtualizace : Podrobný průvodce*. Brno : Computer Press, a.s., 2010. 408 s. ISBN 978-80-251-2676-9.

[18] ROSEN, Rami. *LinuxJournal* [online]. 2006-03-02 [cit. 2011-05-31]. Dostupné z WWW: http://www.linuxjournal.com/article/8909?page=0,1

[19] *Microsoft* [online]. 2006 [cit. 2011-01-31]. Dostupné z WWW: <http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/vir054_wh06.ppt>.

[20] *VMware* [online]. 2007-09-11 [cit. 2011-06-01]. Dostupné z WWW: <http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AMD-V	AMD-Virtualization.
API	Application Programming Interface = rozhraní pro programování aplikací.
CPU	Central Processing Unit – procesor.
C++	Objektově orientovaný jazyk.
EPT	Extended Page Tables.
FAP	Fyzický adresný prostor.
FSB	Front Side Bus – sběrnice.
IVT	Intel Virtualization Technology.
HW	Hardwarové vybavení..
MMU	Memory Management Unit.
MPN	Machine pages.
LAP	Logický adresný prostor.
LPN	Logical Pages – logická paměť.
NPT	Nested Page Tables.
OS	Operační systém.
PPN	Physical Pages – fyzická paměť.
QPI	Quick Path Interface – sběrnice.
RAID	Redundant Array of Independent Disks - vícenásobné diskové pole laciných/nezávislých disk. Je to metoda zabezpečení dat, proti selhání pevného disku.
SVM	Secure Virtual Machine – zkratka pro hardwarovou virtualiazaci od fi AMD.
SW	Softwarové vybavení.
VMCB	Virtual Machine Control Block Strukture – terminologie AMD = VMCS.
VMCS	Virtual Machine Control Strukture.
VM	Virtual Machine – virtuální stroj.
VMM	Virtual Machine Monitor – hypervizor.
VMRUN	Virtual machine Run.
VT-x	VT = virtualization technology, přípona „x“ označuje technologii od fi Intel založenou na procesoru.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH