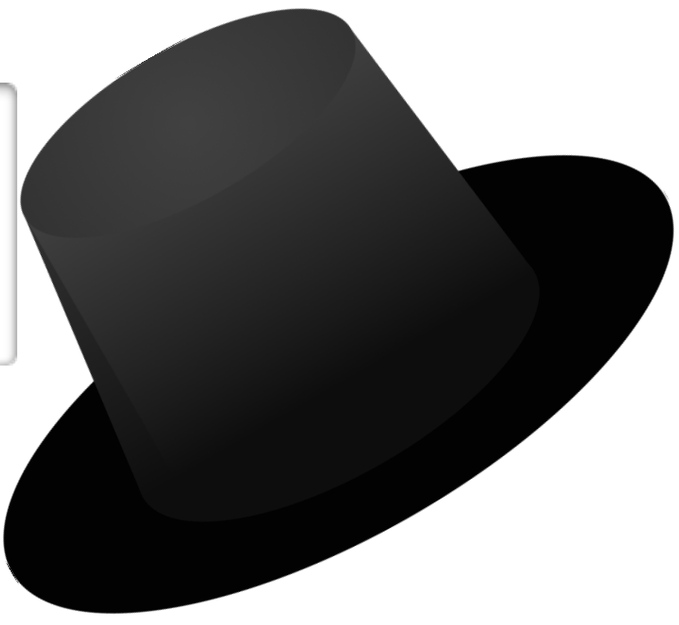
仮想化環境
- 低レイヤからの仮想化環境知識 -

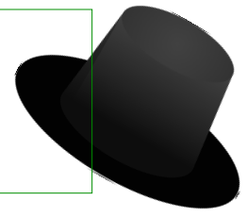
MohikanZ app-infra-sec channel

マサカリ担当 メシテロリスト

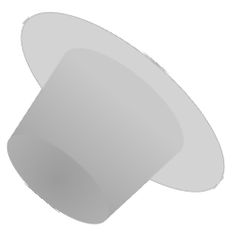
seirios



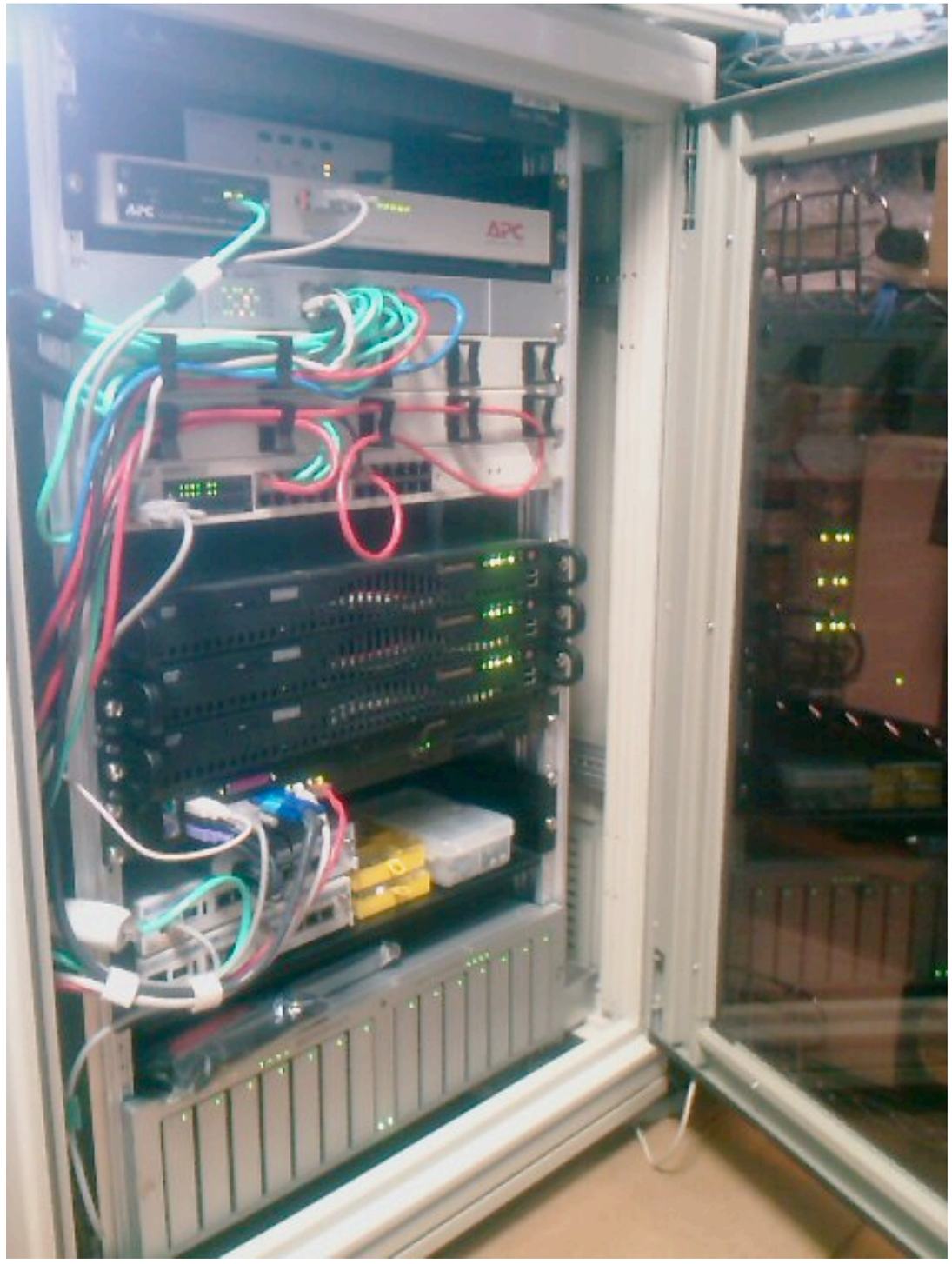
Who am I



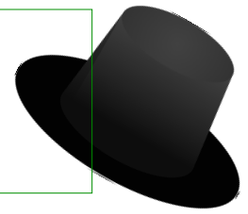
- 略歴
  - 1995年、ISPに入社（いまだにサービス継続しているVenture ISP)
  - 1997年、技術コンサル系会社に転職（上場→上場廃止→海外再上場で存続中）
  - 2001年、MSPに出向（管理職）→役員
  - 2005年、MSPと本体のコンサル部門とISP系会社合併（役員）→（役員）
  - 2010年、セキュリティ会社に転職（一般技術職）
  - 2014年、データセンター会社に転職（一般技術職）
  - 2017年、子会社転籍（一般技術職）　←イマココ
- 活動履歴（ありすぎるのでかいつまんで）
  - 現任：WIDE Project研究員、JPNIC専門家チームメンバー
  - 卒業：JNUG事務局長、IPv6インストール大会主催、他...
  - 執筆等：インターネットルーティングアーキテクチャ（監修）、BGP4（翻訳）、インターネットルーティングプロトコル、BSD magazine、これから始めるIPv6(@IT)...



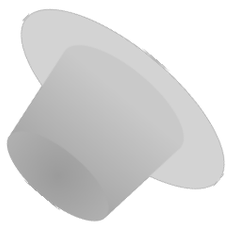
在りし日の我がComputer room



本職は？

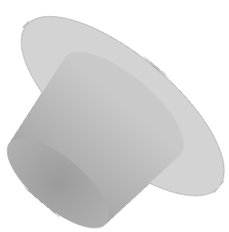


- 本職は... 何でしょう？
  - ISP等BackboneにおけるNetwork Architect、Routing Engineer
  - サーバー系基盤技術者(OS/Middleware/Server daemon）
  - Security Engineer
  - 時には研究者、時には運用者。経営者だったこともあったかな。(CV 増山江威子)
- ある時は、IoT Security関連技術者兼コンサルタント
- またある時は、インフラサービス設計構築技術者
- その実態は、「**MohikanZで鉞を振るうただの老人**」

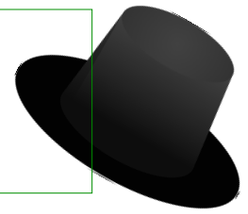


本日のお題

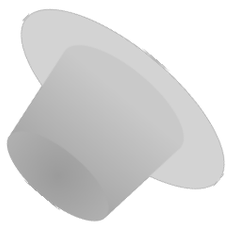
- 仮想化技術基盤 -



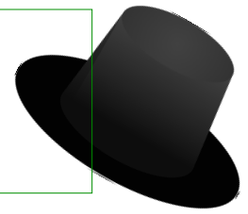
仮想化とは？



- 岩波国語辞典第６版
  - 「事実でないことを仮にそう考えること」
- Computerにおける仮想化とは、「何らかのリソース」を「実在するリソースであるかのように見せる」技術
  - **Virtual Memory**
    - 物理的に実在しているわけではないメモリーを存在しているかのように見せる
    - 手法としては、**swap**などがある
  - **Virtual Disk**
    - 物理的に実装されていないDiskを存在しているかのように見せる
    - 手法としては、**NFS**、**iSCS**など**NAS**系技術がある
  - **Virtual Private Network**
    - 物理的に直接接続しているわけでないNetworkを直接接続されているかのように見せる
    - **IPSec**や**L2TP**、**SoftEther(EoIP)**、**SSLVPN**などがある
  - **Virtual Machine**
    - 物理的に存在しない機器を存在しているかのように見せる
    - **HyperVisor**や**Container**などがある



仮想化の意味



- そもそもの背景
  - PCの性能(CPU)は、ここ30年で「無限と言えるほど」向上した

CPU比較表

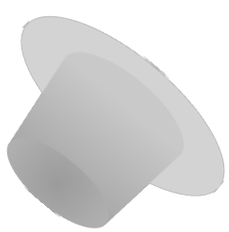
	Z80	Pentium PRO	Core2 Duo
発売	1976年	1995年	2007年
Clock	2.5MHz	200MHz	3GHz
扱えるMemory	64KBytes	4GBytes	10PBytes
演算性能	1MIPS	350MIPS	11,000MIPS

MIPSで測定しても1万倍。浮動小数店演算なども考慮すると100万〜1000万倍

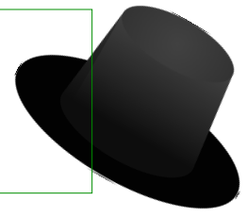
  - Networkの性能はここ15年でせいぜい5000倍

Network環境比較表

	1994	2000	2009
家庭の帯域	2.4〜115.2k	128k〜768k	1.5M〜100M
実効帯域	4.8Kbps	64Kbps	20Mbps
ISPのIX帯域	1.5Mbps	45M〜100Mbps	1Gbps



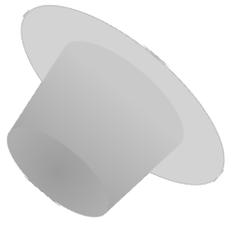
仮想化の意味



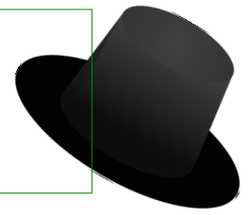
- そもそもの背景 (Cont.)
  - **サーバー性能は十分に要求に応えられる所まできた**
    - Multi Core CPUによる密結合マルチプロセッサアーキテクチャ
    - SMPを用いたMulti CPUアーキテクチャ
  - **利用者は激増**
    - 1997年当時570万人、2000年当時2000万人、2007年当時8200万人
      - その結果、サービスの利用者も大幅に増えた
  - **場所代の高騰**
    - 電力消費量、発熱量（電力消費量に比例）、絶対数値としての回線費用
  - **サービスに対する要求が厳しくなった**
    - サービスに要求される品質
    - サービスで取り扱うデータの重要度

この状況を考えると、今までのやり方でサービスを継続するのは「コスト高」になり、利益が出ない

技術的なブレークスルーが必要

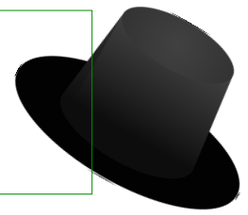


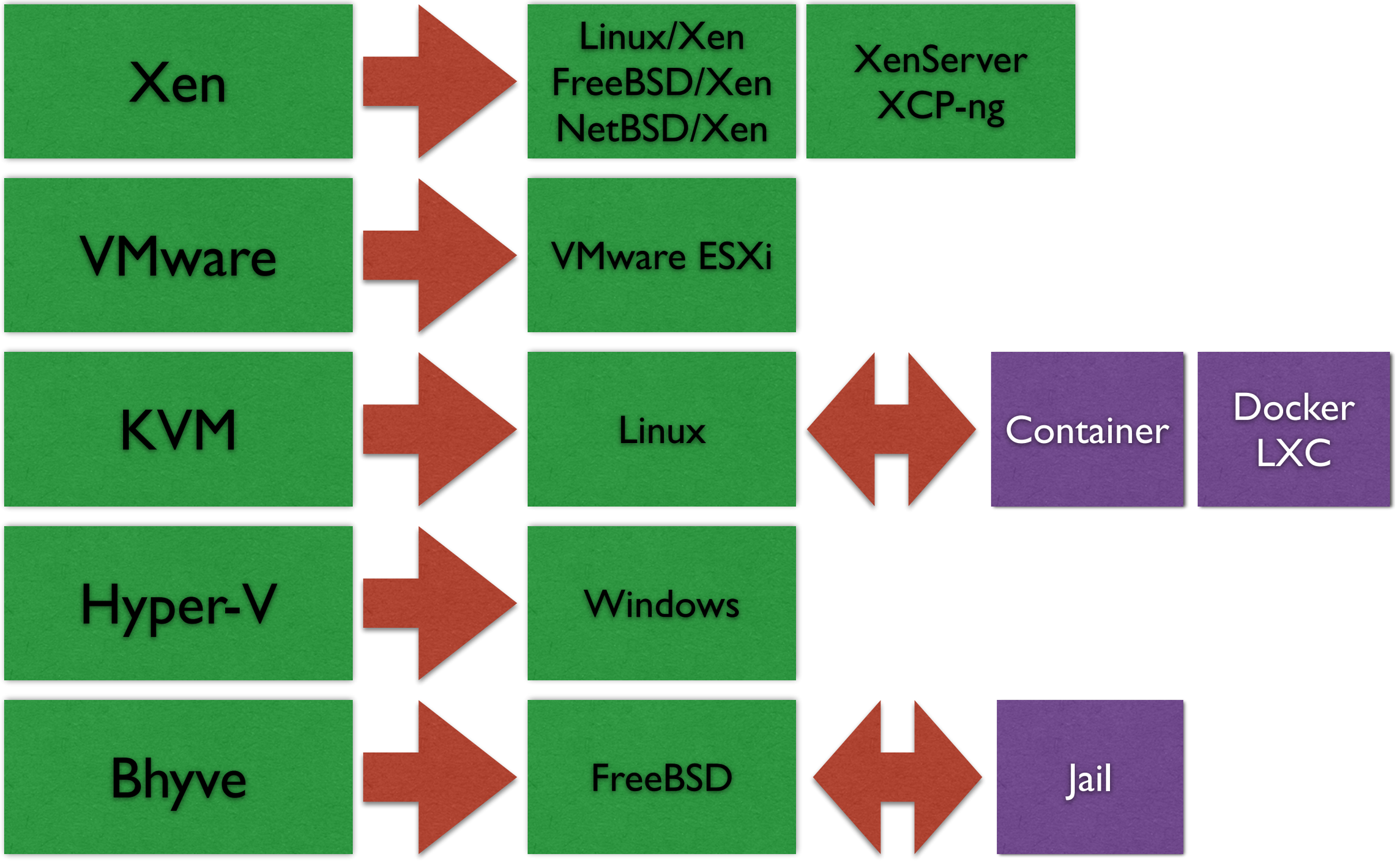
仮想化の手法と歴史



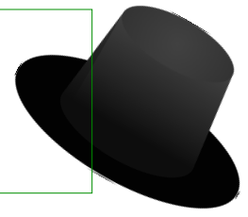
- Serverを仮想化するにはいくつかの手法がある
  - Software Simulatorでプロセスを実行できるようにする
    - Monoなどがこのタイプ
  - Software EmulatorでPC自体を擬似的にEmulateする
    - 初期のQEmuや、今時のPS-Emulator/X68000 Emulatorなどがこれ
  - ハードウェア(主にCPU)の機能を用いて、PC自体を擬似的にEmulateする
    - これが、XenやVMware、Hyper-V、KVM等の手法
- 仮想化小史
  - 仮想マシンの考え方　　: 1972年IBMがリリースしたSystem/370
  - Emulation型仮想化　　: 1999年VMwareリリース(大型コンピュータ→x86へ)
    - Windows95用のアプリをWindowsXPで動かすために大流行
  - HyperVisor型仮想化　　: 2002年VMware(ESX Server)リリース(vmkernel)
  - HyperVisor+ParaVirt　　: 2005年Xenリリース

現代の主要仮想化システム

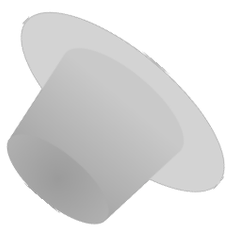




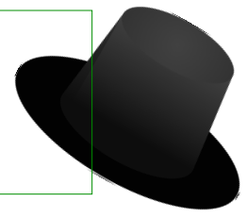
仮想化システムの分類



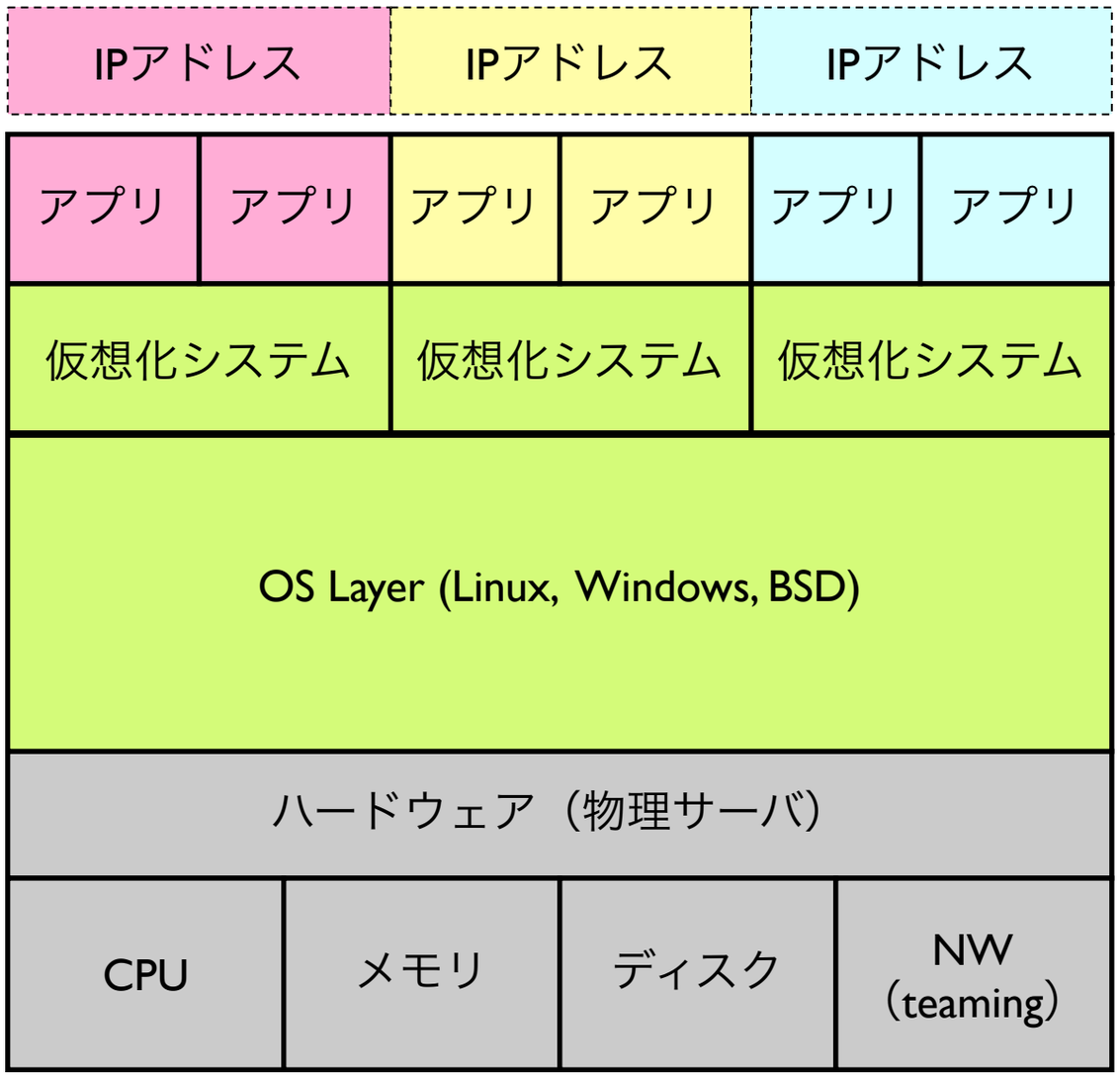
- 仮想化対象による分類
  - **Hardware(PC/Server) の仮想化技術**
    - Hypervisor分離型 : Hypervisor layer と Management layerが分離されている
      - Xen(xen) + domain0 / VMware(vmkernel) + 制御OS
    - Hypervisor/OS一体型 : OSのProcessとして仮想化対象が動作する
      - KVM(linux) / Hyper-V(Windows) / Bhyve(FreeBSD)
  - **OSより上の層の仮想化技術**
    - jail (FreeBSD) : chroot系。user name spaceは分離されていない
    - container : docker/LXC等。user name spaceが分離されている



仮想システムの構成比較

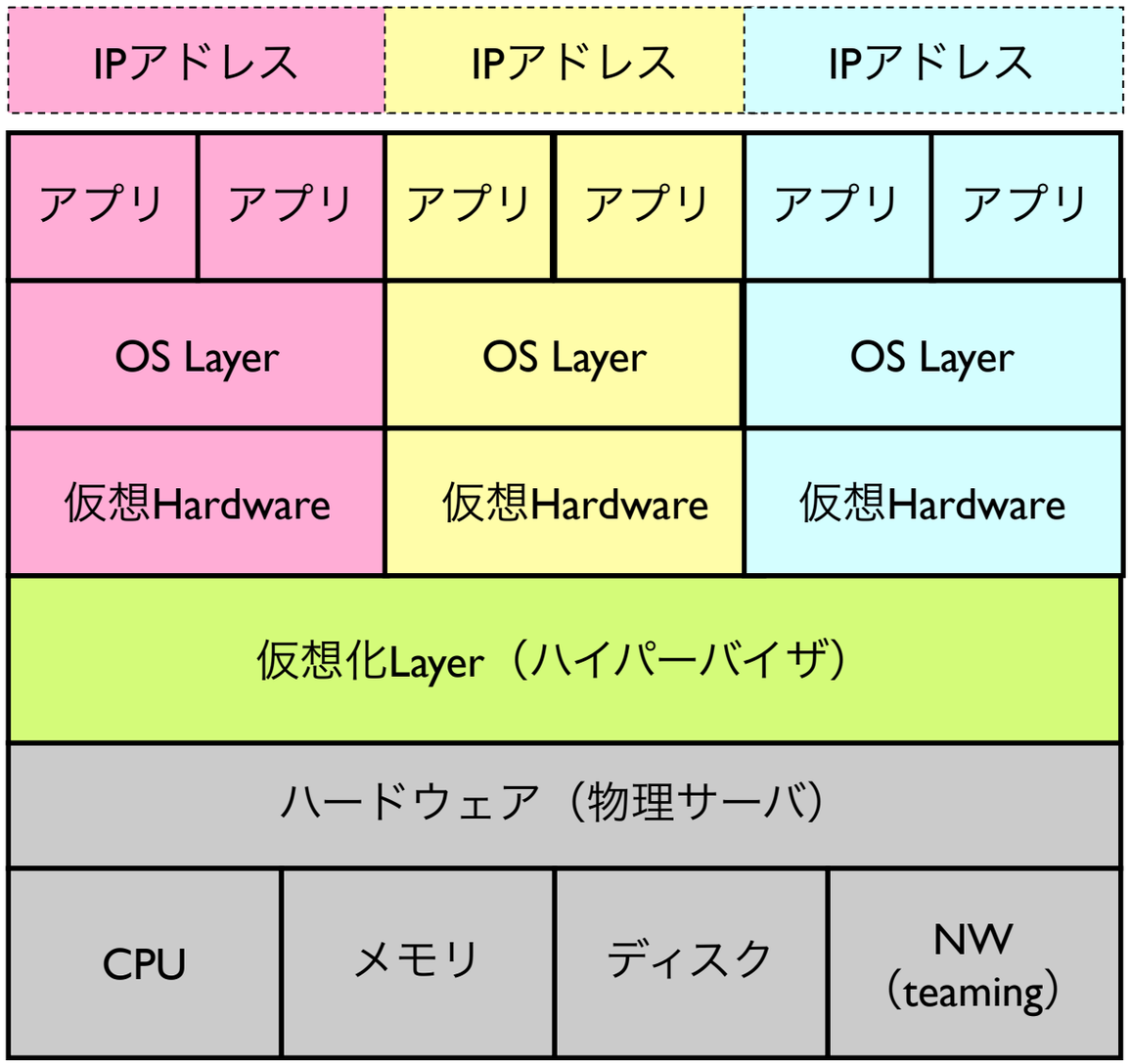


docker・LXC・jail

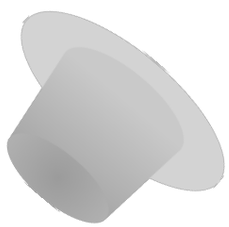


OS Layerは共通

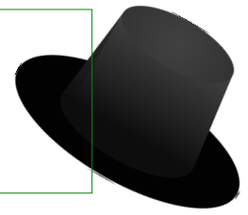
VmwareESX・Xen・Hyper-V・KVM・BitVisor



基本的に全て仮想化

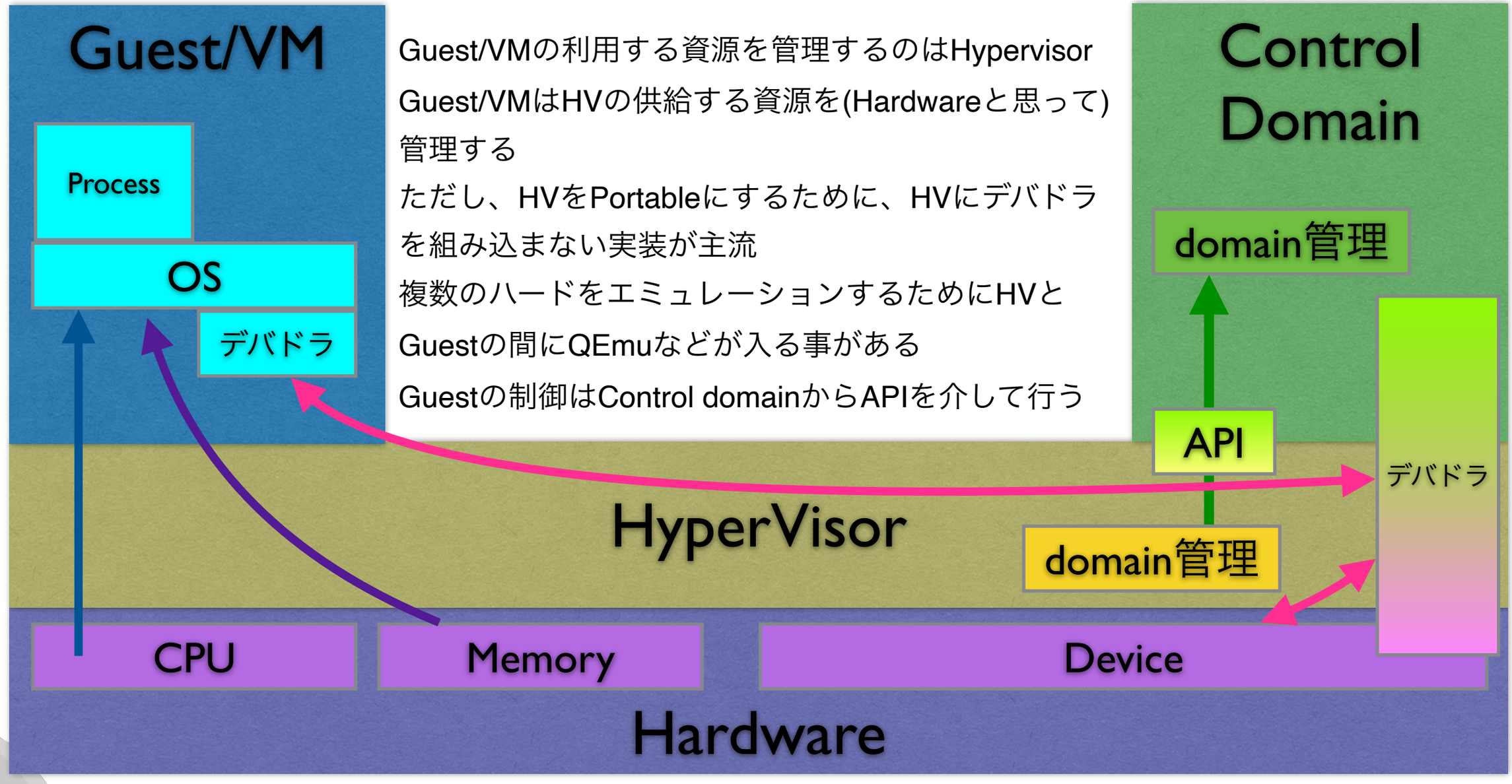


Xen(Hyper-V/VMwareも)の構造

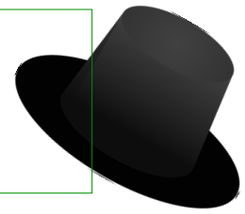


この種の仮想化システム(hypervisor型)は、

HypervisorがHardwareとOSの仲立ちをする



KVM/Bhyveの構造



この種の仮想化システム(OS Include型)は、

OSがControl DomainとHypervisorを兼ねている

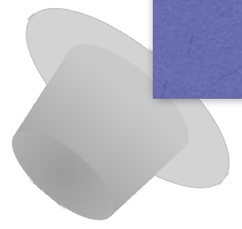
Guest/VMの利用する資源を管理するのはHostOS

割り込み制御などが減るので、効率は良い

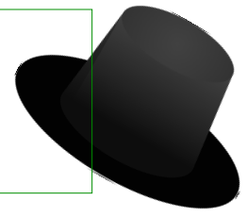
複数のハードをエミュレーションするためにHVと

Guestの間にQEmuなどが入る事がある

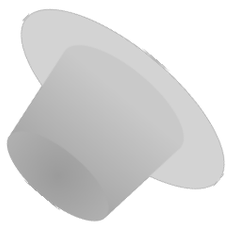
Guestの制御はHostOS上で動作するプログラムを利用する事で行う



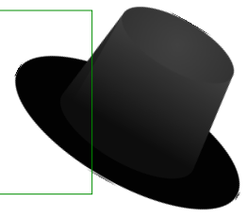
今時のHV型Cloud(IaaS)



- AWSは、昔はxenを利用していた(今も)が、最近ではKVMが追加された
  - Hardware Emulationは不明だが、おそらくQEmuと思われる
- GCPはどうやらKVMに見える
  - Hardware Emulationは独自実装
- AzureはHyper-Vのようなものを利用(Windows Azure kernel)
  - Hardware Emulationは非公開



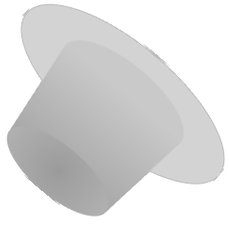
PC97アーキテクチャにおけるBoot



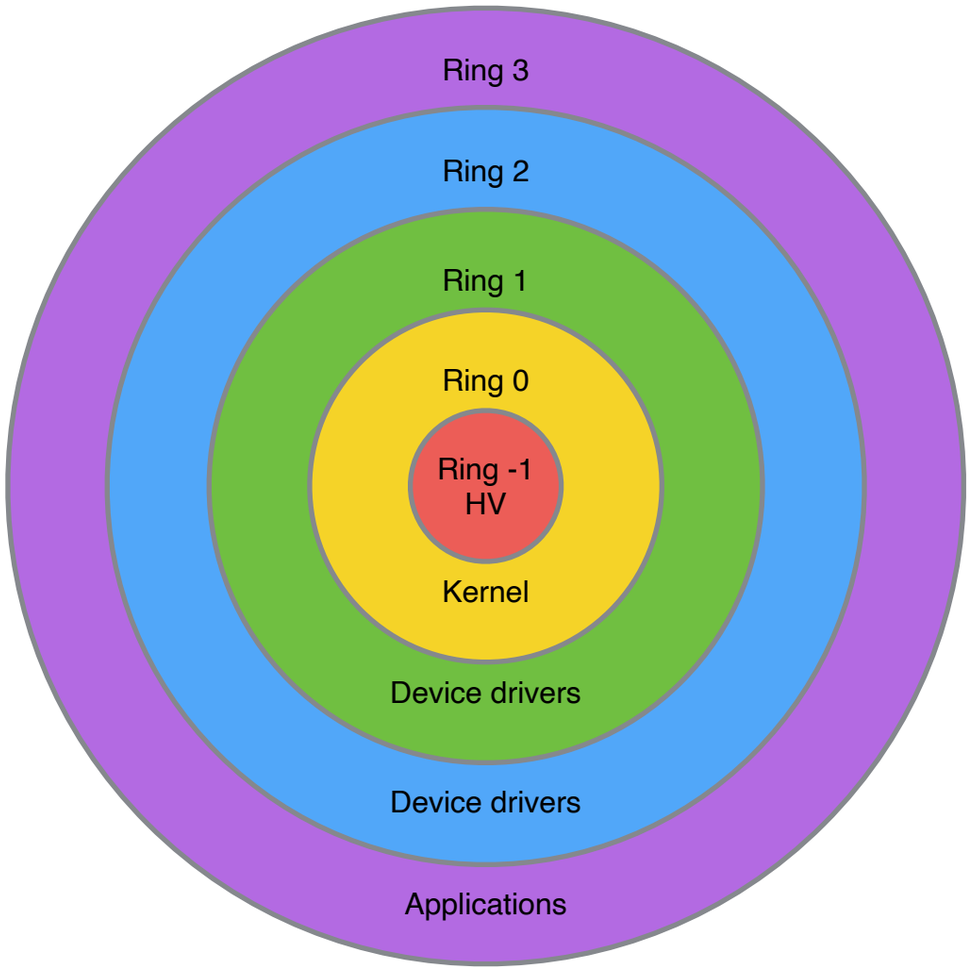
- Hardwareに電源投入
- CPUがROM(Flash)に記録されているBIOS/EFIを読み込み、実行する
- BIOS/EFIが最小限の各種初期化を行なう
- 起動ドライブを探す
- 起動ドライブからMBRを読み込む
- MBRが起動する領域を探す
- MBRがboot sectorに記述されたIPLを読み込み、実行する
- IPLがファイルシステムを判別する
- IPLがOS固有の最初のファイル(kernelなど)を読み込む
- kernelなどがOS固有の手続きを実行する

仮想Emulatorはこれらを全て実行できなければならない

そのためには、MemoryやDiskなどCPU, I/Oなどを全てPC97アーキテクチャに合わせて供給できなければならない

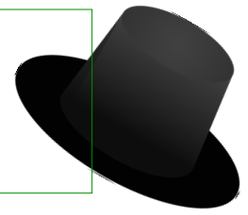


CPUの権限とRing Architecture



- CPUに置ける権限分離は Ring Protectionによって行われる
- 元々の(86系)Ring構造は左の通り
  - Ring 0をSuperVisor modeと呼ぶ
- もともとはこれで十分だった
- 仮想化を行うにあたって、OSのKernelよりも高い権限の動作モードが必要になった
  - この動作モードを供給するのがIntel VTおよびAMD-V
- Ring 0よりも権限の高いRing -1を供給する事で、HVを分離する

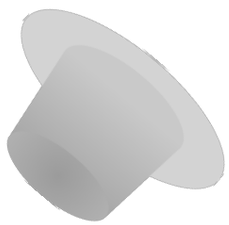
皆が使っている仮想環境とは



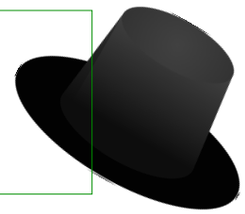
- 利用者は、遠隔地にあるPCの感覚でCloudサービスを利用している
  - 利用者にとっては、**仮想環境なのか物理的存在なのかは関係がない**
  - 気にするのは「**使い勝手**」と「**性能**」だけ
- 現在のCloud環境は、ほぼ「仮想環境」と言って過言ではない
  - 一部DBなど “High Performance” を要求される局面でのみ物理サーバーを利用
- 更に言えば、いわゆるConsoleすら利用していない
  - **APIやGUIは触る**だろうが、BIOS設定とかしないでしょ？
  - つか、そもそもBIOSって知っている？
  - 仮想サーバーに関して言えば、「電源オフ」とか「電源オン」の感覚もない
    - つまり、Boot Sequenceとか正しく理解していないよね

今のCloud環境は「正しく抽象化されたサーバー供給」環境

インフラ・ハードウェアの概念が不要



クラウド移行のモチベ



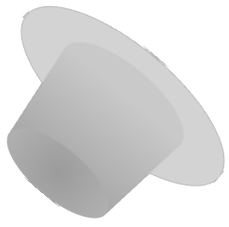
- 仮想環境にする事で得たいもの
  - ハードウェア性能を使い切る（多少無駄であっても）
  - 安定性をあげる
  - 拡張性をあげる
  - コストを下げる（本来は結果。現実には要求）
  - 簡単な管理と簡単なデプロイ

と、いろいろありそうに見えるが

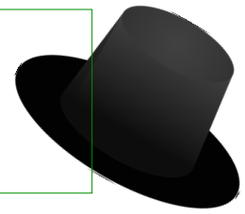
「インフラなんか面倒見たくない」

「すぐに焼却される資産なんて持ちたくない」

が本音なんじゃないの？



クラウドを支える技術



- Server群管理技術
  - VM管理I/F
  - NW管理I/F(vLAN管理など)
  - Pooling
  - VM Migration(Cold/Live)
  - Storage Migration(Cold/Live)
  - Priority Control
- Storage管理技術
  - 多重化技術(RAID/Clustering)
  - バックアップ技術
  - 転送性能向上技術
  - 故障対応技術

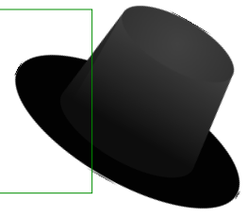
- Network管理技術
  - vLAN Tagging
  - Tag in Tagging
  - Dynamic Configuration
  - Traffic Monitoring
  - Bandwidth Controlling
  - Traffic Shaping
  - Flow Sampling

必要となる技術が多く

相互関連が強いが

利用者に見えるのはほんの一部

オンプレ仮想化環境の勘所

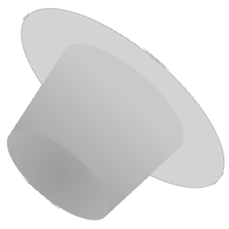


- 基本構成
  - 1 PoolあたりのServer数やStorage構成
  - Storage ←→Serverの通信帯域とStorageのIOPS
- Capacity Planning
  - 1 HostあたりのGuest数の（CPU core vs VM数）
  - 1 GuestあたりのMemory量（物理限界とSwap）
  - 1 PoolあたりのNetwork Traffic量（物理限界論理限界）
- Network Planning
  - vLAN構成の検討
  - Traffic Controlの必要性と手法
  - 管理ネットワークの構成
- 運用・管理
  - 死活、リソース消費量モニタリング
  - 性能モニタリング
  - 自動化とセキュリティ
  - 管理者と利用者の権限分離
  - 管理システムへのアクセス
  - Guestを最大効率で動作させるためのリソース分配
  - Overheadの最小化(Disk/CPU/PV vs FV/VT-d/SR-IOV)
  - Guest状態の監視

まとまりがつきませんが、とりあえず今日のところはこのくらいで

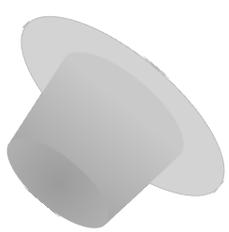
ご静聴

ありがとうございました

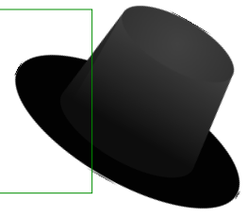


おまけ

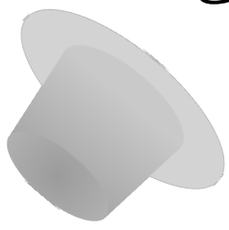
CPU Virtual Memory History



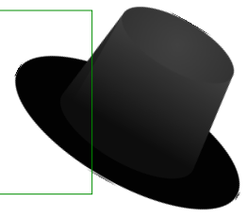
Virtual Memory History



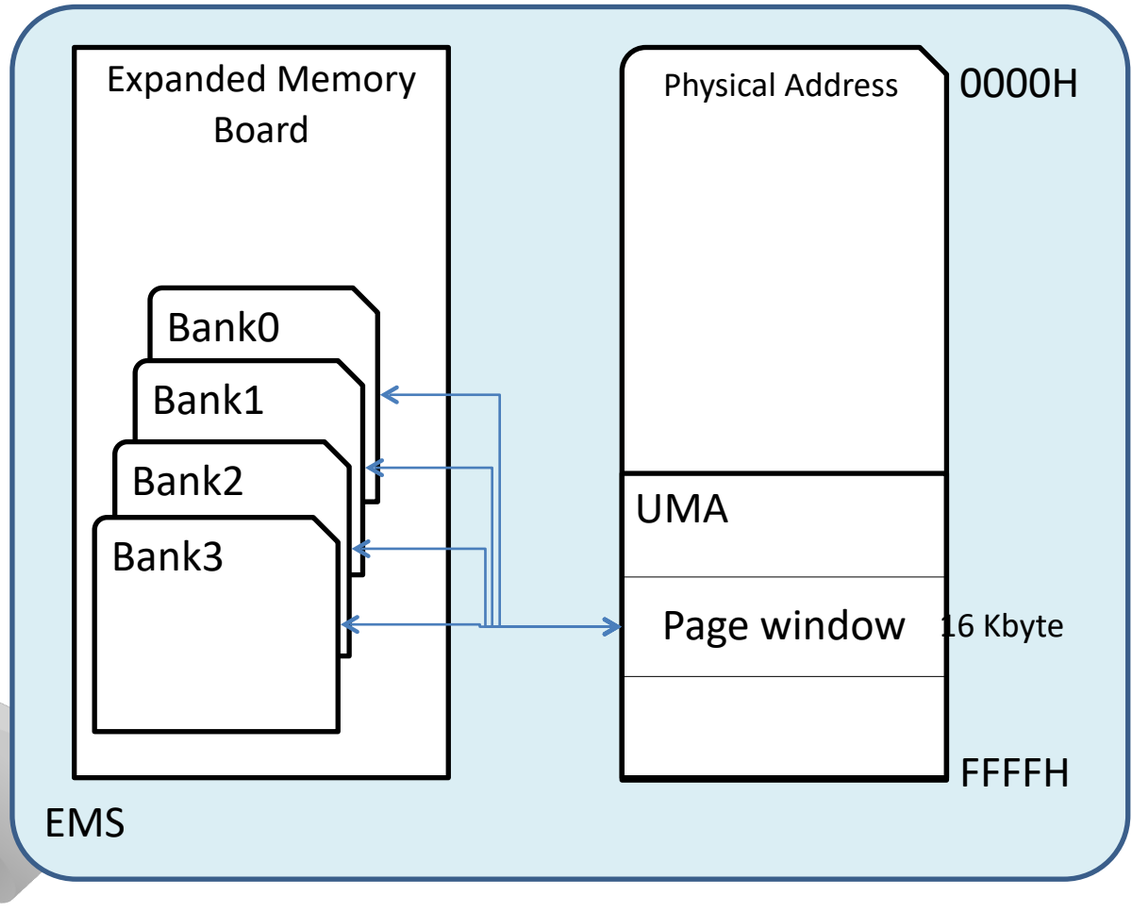
- **CPUが取り扱えるメモリー空間のサイズ**と**入手可能なメモリーチップのサイズ**はその**時代時代で必ずしも一致しない。**
- **CPUがアドレッシング可能な空間より大きいメモリーを取り付けたい場合もある**し、**CPUの広大なメモリー空間内に容量や速度の異なるROM/RAMを遍在的に配置したい場合もある。**
- CPU上で実行したい**プログラムサイズは肥大化の一途**を辿り、**実装可能なメモリー量を凌ぐサイズのプログラムを並行動作させる必要**に迫られる。
- アドレス空間の肥大化により**リロケータブルオブジェクト型CPUのISAが非現実的**になり、**フルリロケータタブルなCPUが終焉**を迎える。
- マルチタスクOSの普及に伴い、**単一オブジェクトの複数同時実行時等に論理的なアドレス空間を扱いたい。**
- ハイパーバイザーの普及に伴い、**ゲストOS毎に物理メモリに見立てた論理メモリを定義**したい。
- 等々のニーズが生まれ、それらを実現するべく**ソフトウェアプログラマ達によりソフトウェア化**されていく一方で、それらを**CPUメーカがハードウェアで実現**して**高速化・低コスト化**して**ソフトウェア実装からハードウェア実装へ移行していくというマッチポンプ**が進んでいきます。

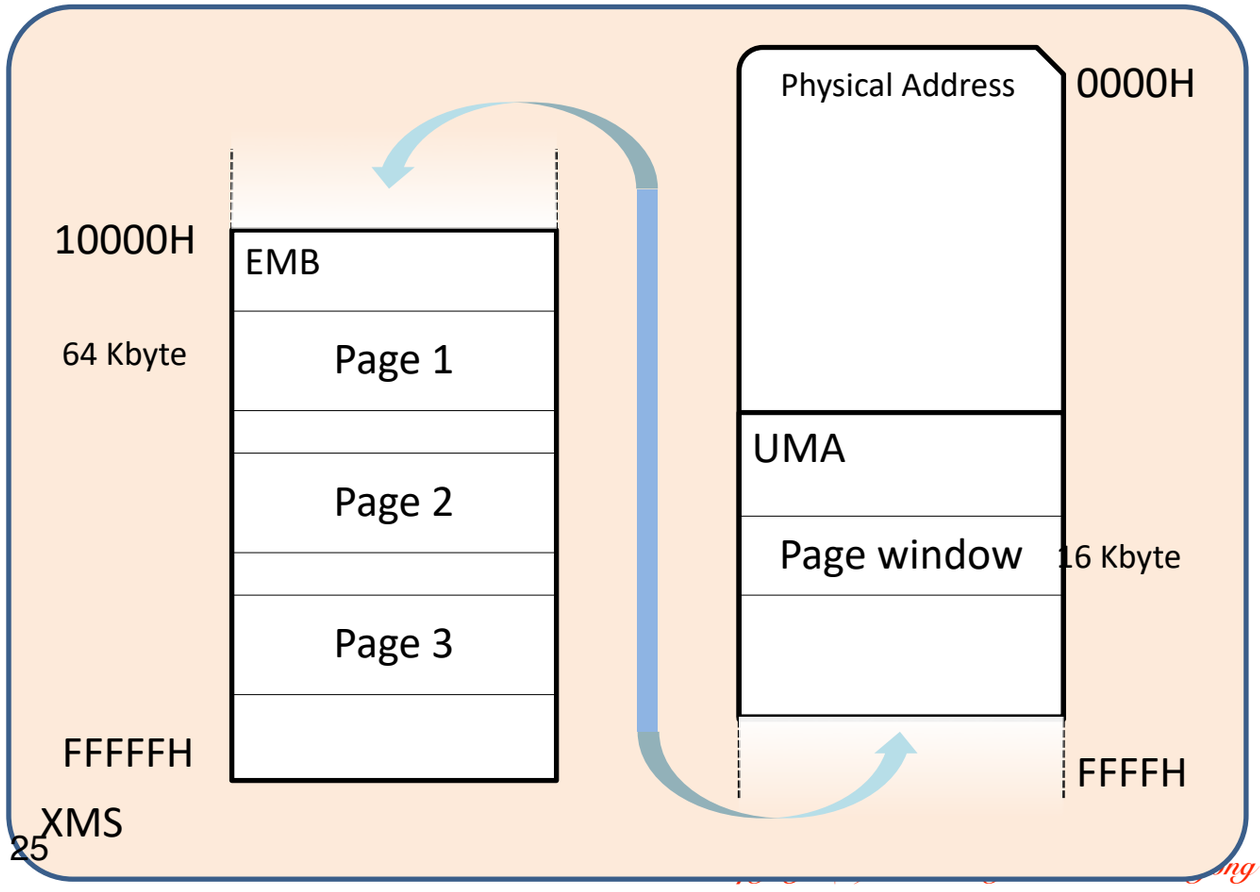


バンクメモリー（仮想メモリ前史）

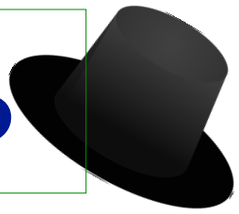


- i8080,Z80,MC6800等の8bit CPUのアドレスバスは16bit幅で有った為、アドレッシング可能なメモリー空間は64Kbytesであったが、より広いメモリ空間を求めて、i8086,MC68000等の更にアドレス空間の広いCPUが登場してきた
- アドレスが広がった分、プログラムの肥大化や常駐タイプのプログラムの登場により、640Kbytesでは不足する場面も多くなり、より広いアドレス空間が求められるようになった
- メモリーアドレス空間の一部を切り替え可能にし、ソフトウェア上から任意のバンクメモリーを切り替えられるようにして、より広いメモリ空間を提供した
- バンク切り替えを行うプログラム自身はバンクメモリー上には置けない事や、複数バンクのメモリー同時参照が出来ない等、通常のメモリーと区別して使用する必要があった
- i8086でも使用できるIntel/Microsoft/Lotus社が提唱したEMS等や、i80286以上のプロテクトモードで拡張されたメモリを使用できるXMS等が代表的
- i80286以降になっても、メモリ空間の広いi8086としてMSDOSで利用することが長く続き、HMA/XMS/EMB/UMA/UMB等の3文字略語が乱立し、i80486の時代になるとA20M機能等が登場し、益々利用者を混乱させることとなる



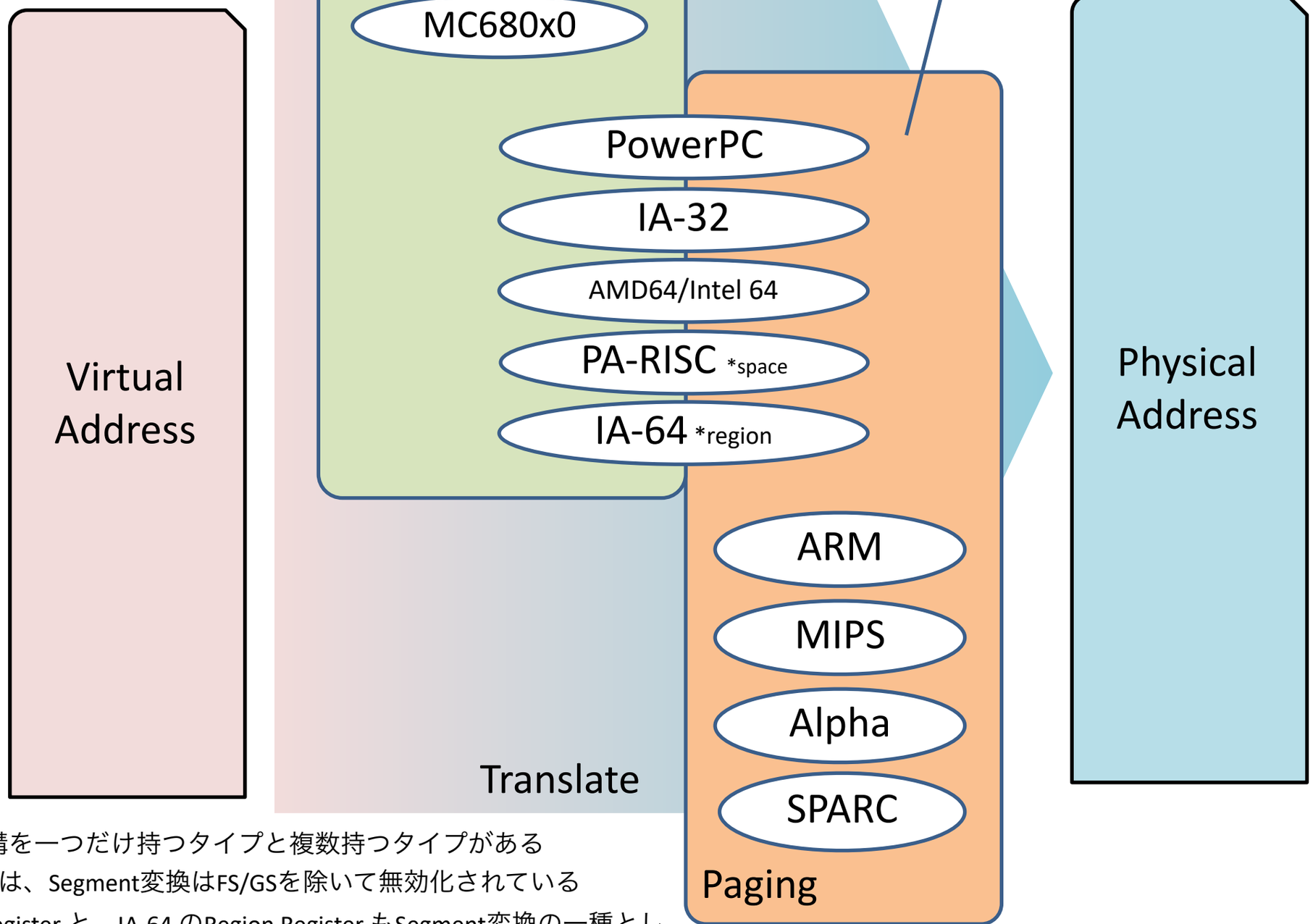


仮想メモリはCPUアーキテクチャの数だけある



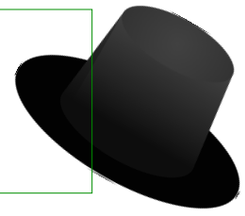
メモリーを可変長なセグメント単位に仮想メモリーに割り当てることが可能
メモリーマネージメントユニット(MMU)はCPU内蔵若しくは周辺ファミリーチップとして提供される

固定長なメモリー管理単位であるページを処理するメモリマネージメントユニット
4Kbytes/pageのものが主流

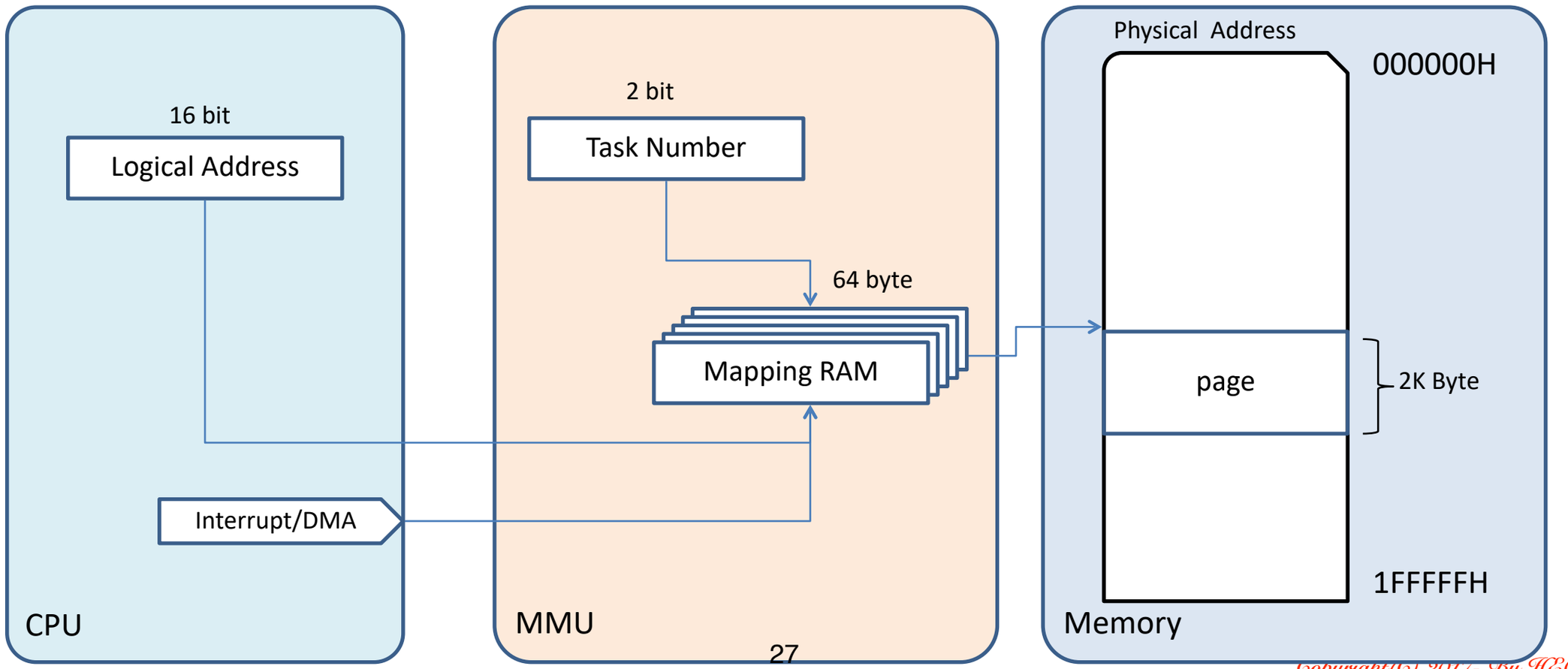


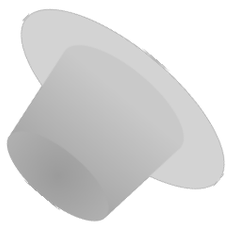
- アドレス変換機構を一つだけ持つタイプと複数持つタイプがある
- AMD64/Intel 64では、Segment変換はFS/GSを除いて無効化されている
- PA-RISCのSpace Register と、IA-64 のRegion Register もSegment変換の一種として記載している

MC6809+MC6829 （8bitでは唯一）

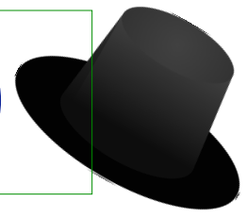


- MC6809本体には仮想メモリの支援ハードウェアは全く無く、メモリ空間も他の8 bit CPUと同様の16bit (64Kbyte)
- ４タスク固定のタスク管理型メモリ管理機構を持つMMU(MC6829：1979年発表)が利用可能
- 8bit CPUでは唯一の21 bit (2Mbyte)の広いアドレス空間を利用可能
- MC6809 CPU自体に保護機能は無いが、MMU側でシステムタスクとユーザタスクの管理が出来た
- CPUから出力されるBus Available, Bus Status Pinにより、TASK0 (SYSTEM), TASK1(DMA), TASK2(User0), TASK3(User1)に割り当てるメモリを個別に管理可能
- Logical Address は 先頭アドレスから 2K Byte 単位に 32個のマッピングメモリを持ち、任意のPhysical Addressへマッピングする事が出来る
- MC6829は8個までカスケード接続が可能で、最大32タスクまで拡張可能
- タスク番号そのものがアドレス変換テーブルのインデックスになっている静的な実装であり、マッピング外のメモリ参照によるリスタート等の機能も無いが、当時としては画期的なOS9が開発され、実際に利用可能であった

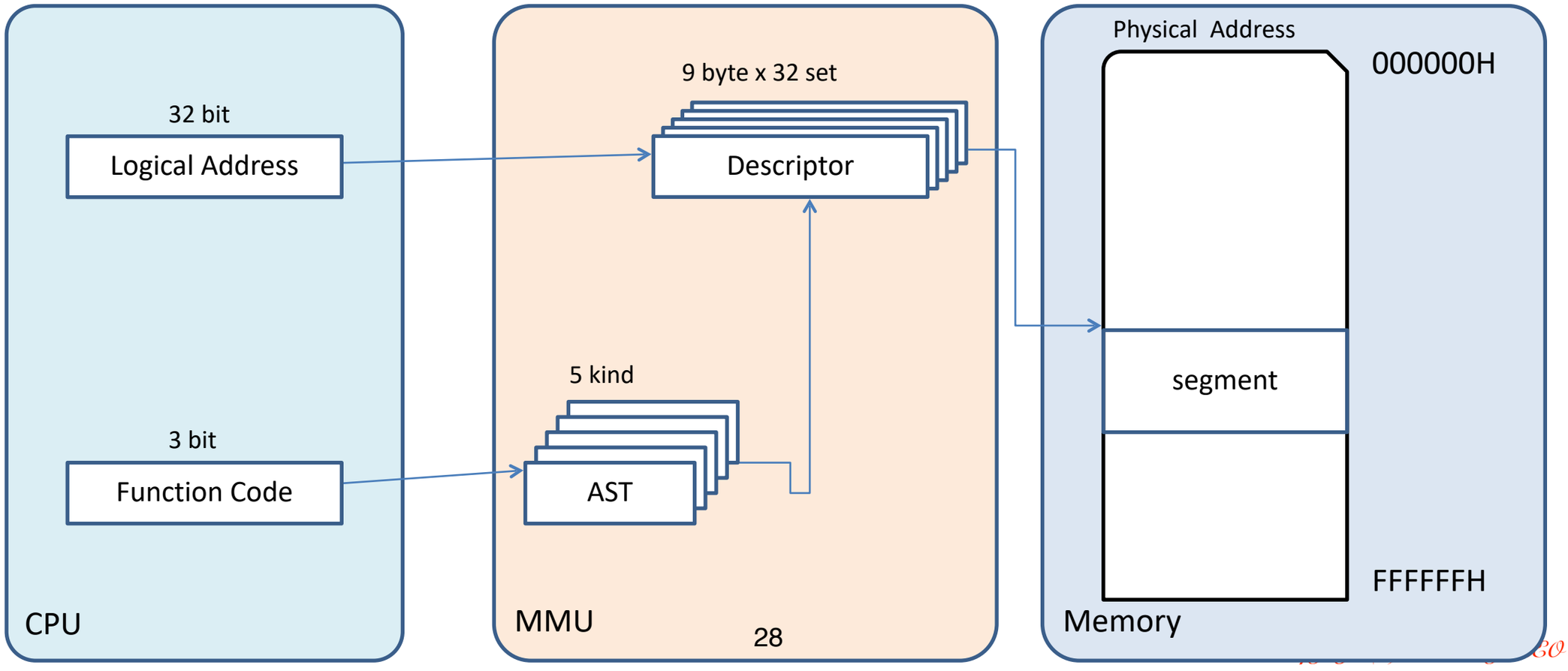


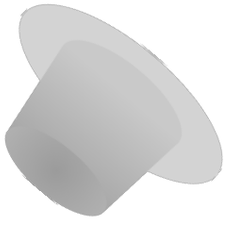


MC680x0+MC68451 （MMUは別売）

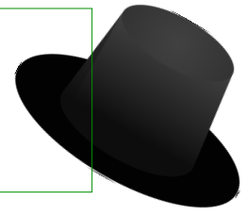


- リニアアドレッシング可能な24 bit (16Mbyte)の広いアドレス空間を持つ、1979年発表
- 特権モードとユーザモードの分離により、メモリ空間も特権・ユーザ空間を別個に管理可能になり、OSレベルでのユーザ/OS特権メモリの保護が出来るようになった
- 最大32セグメントの可変長のメモリ管理単位を採用したMMU(M68451)が利用可能
- CPUから出力されるFunction Codeにより、Address Space Tableの該当するエントリをDescriptorに記述する事により、選択されたSegmentのアクセス権(Supervisor Program, Supervisor Data, User Program, User Data, Interrupt Acknowledge) を個別に定義可能となっている。
- メモリ空間へのアクセス時にバス上でエラー(Bus Error)が発生した際に特権例外を発生させ、メモリバス上に読み書き可能な状態のメモリセルが無い事を知る事が出来る
- MC68000ではBus Error発生時に例外発生元となった命令コードアドレスが保存されていない為、Demand Pagingは実装できなかった
- MC68010はBus Error発生時に現在のバスサイクルを中断し、OSが正しい処理をした後に、バスサイクルを再試行し、例外発生元となった命令コードを再実行する機能が追加され、Demand Pagingの実現に必要な所謂仮想メモリシステムの実装が可能になった

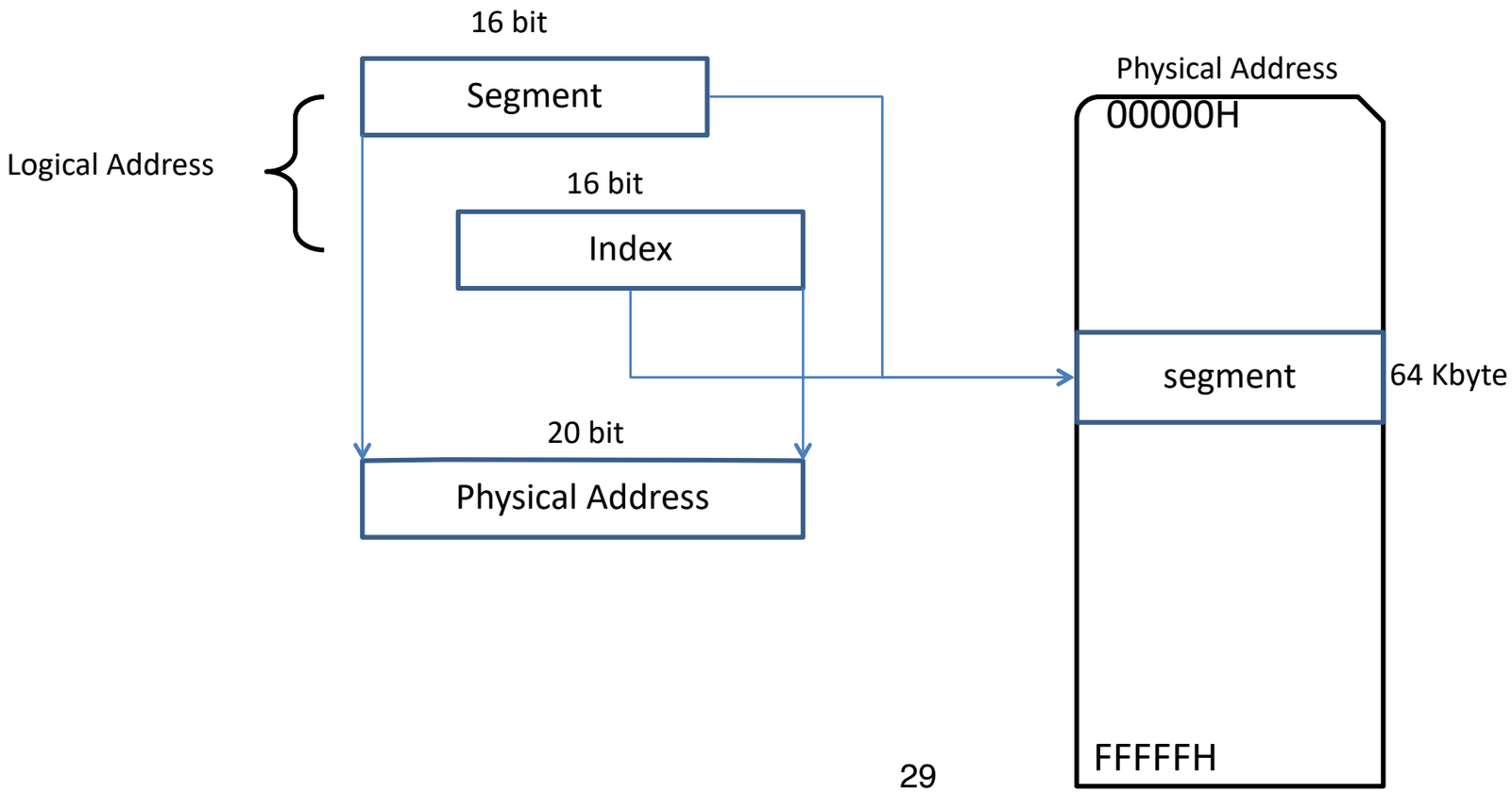


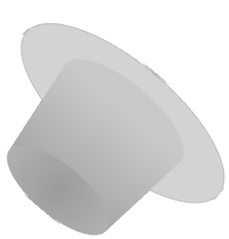


i8086（互換性という長い道のり）

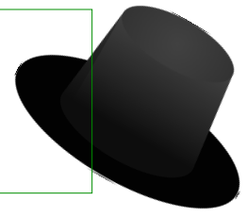


- 大ヒットしたi8080とのソースコードレベルでの互換性を維持しつつ、レジスタ幅を16bit化した、1979年発表
- Segment Registerを新たに導入し、 Index Registerと併せて20 bit (1Mbyte)のアドレス空間を持つ (Physical Address ＝ Segment ×16 ＋ Index)
- i8080で蓄積されたソフトウェア資産を利用可能にしつつアドレス空間拡張する手法論としてセグメントレジスタが導入された
- 当時は1プログラム当たり64Kbyteのメモリー空間は十分に広いと考えられていた時代で、Segmentの概念はマルチタスク用のメモリ分割手法として導入されたと思われ、Physical Addressを命令コード側からの参照時に抽象化する事が目的では無かったと思われる
- その後のプログラム肥大化に伴い、プログラム内でSegment Registerを動的に操作し、64Kbyte以上のプログラムやデータを取り扱うものが登場し、プログラムはスパゲッティ化していくこととなる
- Logical Addressの登場により、プログラムが認識しているメモリアドレスが必ずしもメモリ上のPhysical Addressと一致するものでは無くなり、以降のメモリシステムは大きな変革期を迎えることになる

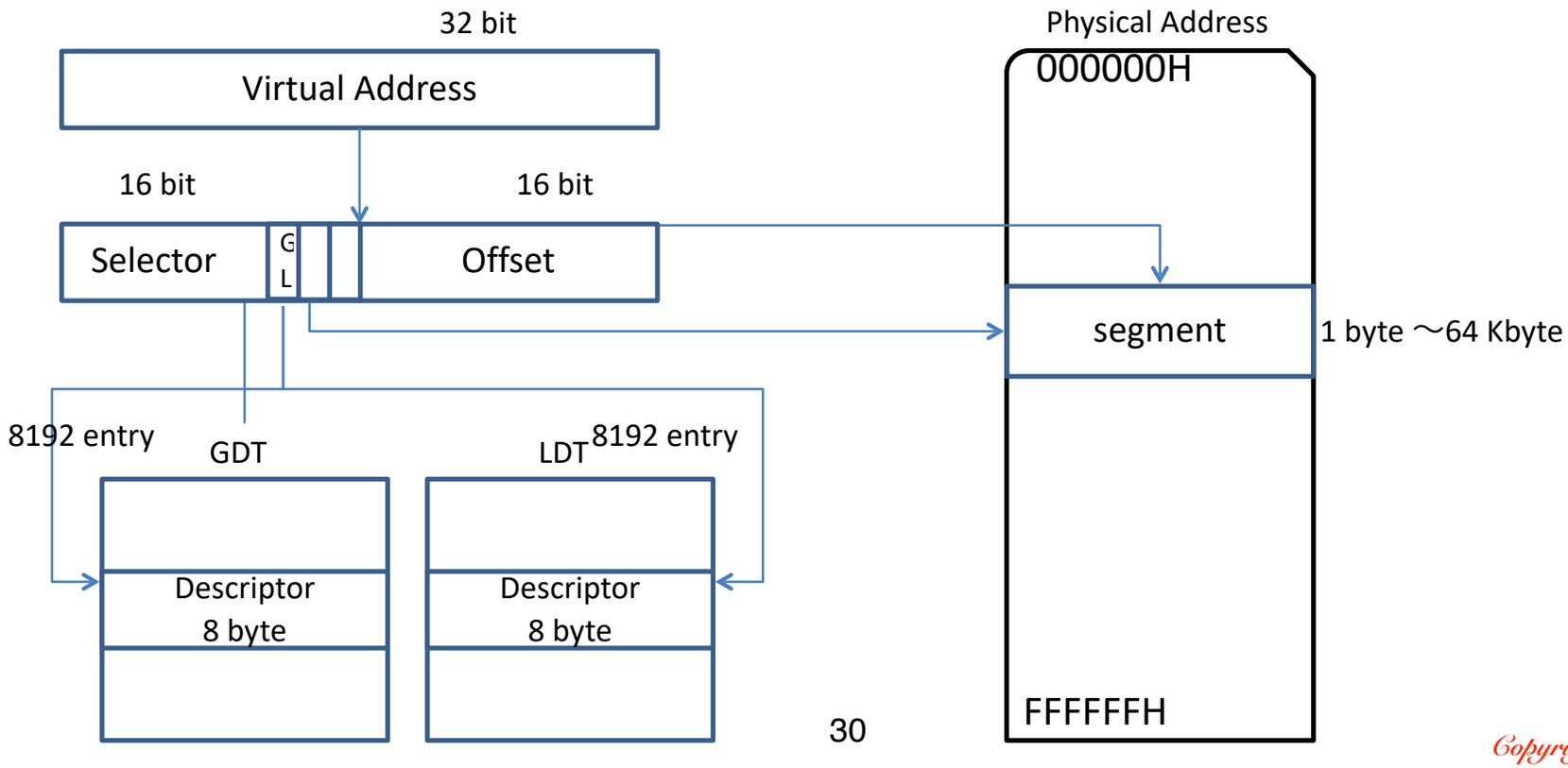


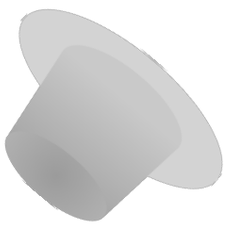


i80286（後方互換性と言う罠）

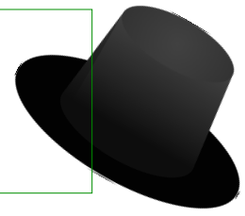


- Intelのロードマップでは32bit CPUでは従来のアーキテクチャを一新して、iAPX432という新設計のアーキテクチャに移行する計画になっていた為、i80286はi8080,i8086との互換性を保ちつつ保護モード等の近代的OSに必要な機能を追加するも、後の拡張性等は余り考慮された形跡が無い（最近も聞いたような気のする話Itan(ry
- 従来のi8086の動作モードをリアルモードとして互換性を保ち、プロセスやメモリを保護する事ができるプロテクトモードを新たに追加
- プロテクトモードでは24bit (16Mbyte)のPhysical Address空間を持つ (リアルモードは20 bit (1Mbyte)のPhysical Address空間)
- Segment RegisterをDescriptor Table を参照するSegment Selectorとして再定義し、 後方互換性を保ちつつ Index Registerと併せて実効長30 bit (1GByte)のVirtual Address空間を持たせた
- アクセス権限を階層化し4つの権限レベル(Ring 0〜3)を使用できる （Motorola等のライバルは2レベルのみ)
- Demand Pagingを実現する為に新たな例外（segment not present fault）を追加し、例外発生元の命令から再実行する事ができるようになった
- プログラムから見えているアドレスは、物理メモリに存在しないアドレスを実際に参照する可能性があるという事から、プロテクトモードではLogical AddressではなくVirtual Address と改称された
- 発売当時はまだMSDOSでの利用が大半であった為、新たに追加されたプロテクトモードを活用したOSは中々登場せず、高速なi8086として利用されていた
- プログラムの肥大化により、Real Mode OSであるMSDOSが取り扱えるConventional Memory では不足となり、Dos Extender, XMS等のProtect Mode で取り扱える1MB以上のメモリをMSDOS上から参照できる方式が多数開発された
- プロテクトモードを活用したOSはWindows 3.xやOS/2 1.0等まで更に待つことになる
- 1982年発表

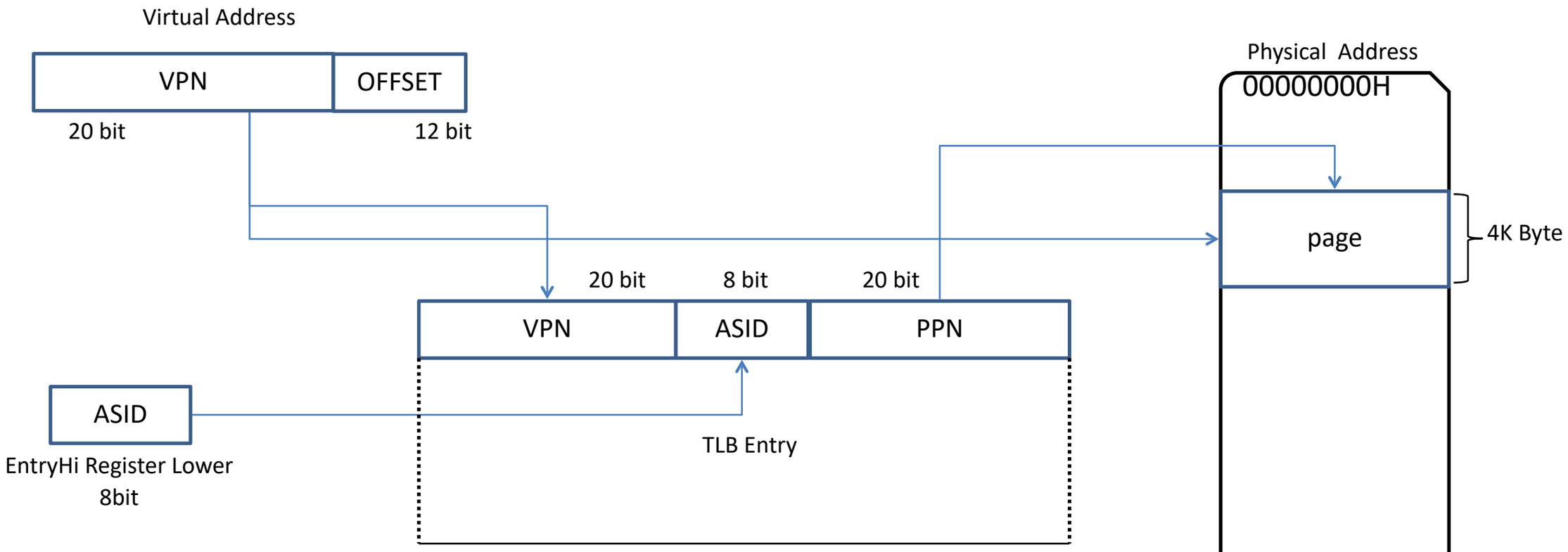




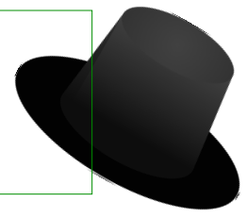
MIPS （RISCの先駆者）



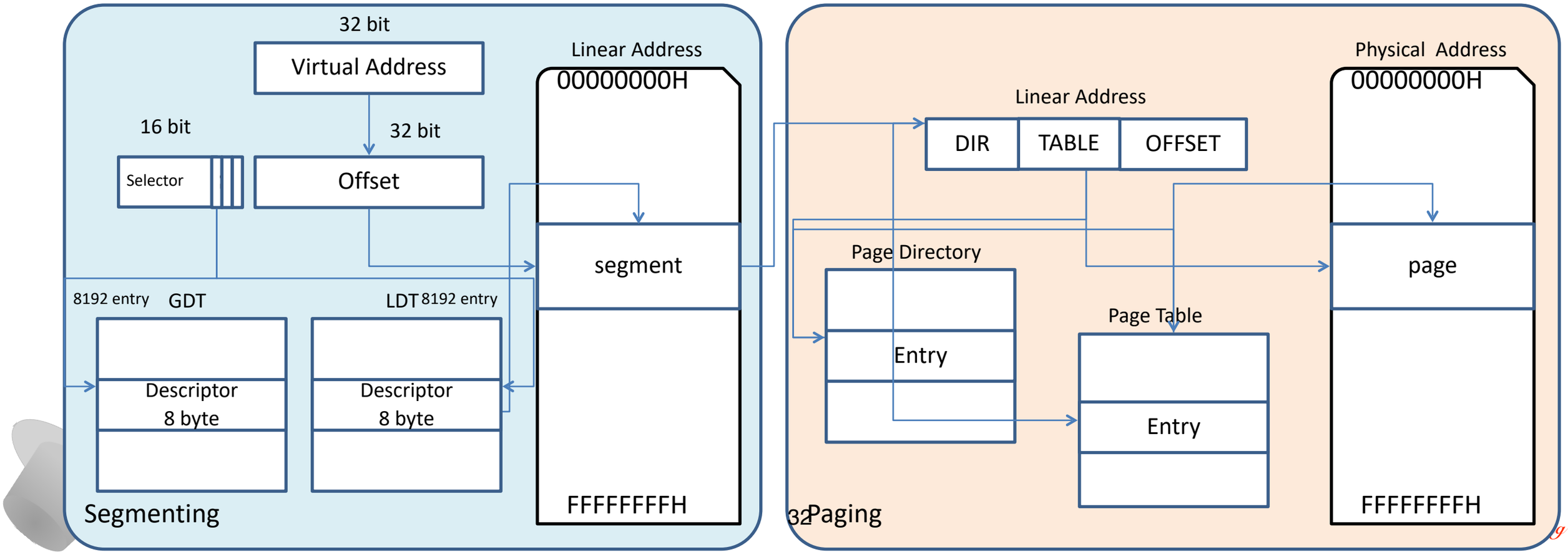
- Motorola/Intelの提唱してきたSegment管理とは異なり、固定長のmemory pageを管理単位とし、page alignmentがpage size の整数倍に制限されている
- TLB (Translation Lookaside Buffer) は、Virtual AddressとPhysical Address の高速変換を目的としたCAM (Content Addressable Memory) として実装されている
- Segmentと異なり、TLBがアドレス参照用のキャッシュとして動作し、 TLBを逐次探索して、Instruction（命令コード）が参照したいVirtual Addressを持ったTLBが既に生成済みであれば、TLB Hitし、matchしたテーブルのPPN(Physical Page Number)が示すPhysical Memoryが確定する。未生成であれば、TLB Missとなり、TLB Walkと呼ばれるpage table searchをおこなって、空きpageをPPNとしたTLBエントリが新たに生成される
- TLB Walkには、ソフトウェアで処理するタイプと、ハードウェアで処理するタイプがある
- ASIDが8bitしかないため、そのままでは最大256 task/process しか管理出来ないが、TLBは個別のタスク/プロセスを弁別する為に、ASID (Address Space Identifier)を持っていてContext Switch時の負荷を低減することが可能
- MIPSIで実装された当初は、近代的OSの要件としての仮想メモリシステム用途というよりは、ROM/RAM,SRAM/DRAMの区別や速度の異なるメモリサブシステムを命令コード側から透過的に(Transparent)にアクセスする用途が主だったように思われる
- MIPSに限らず、仮想メモリシステムにpage管理を導入しているCPUアーキテクチャでは何等かのTLB Walkerが実装されており、仮想メモリシステムはCPUアーキテクチャの数だけあると言えるのと同様に、TLB Walker もCPUアーキテクチャの数だけあると言ってもいい
- MIPS R2000 1985年発表、R3000 1988年発表



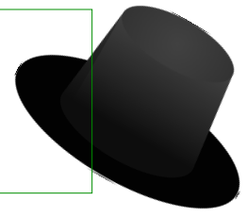
IA-32 （i80386以降）



- i8086,i80186,i80286と迷走してきたセグメントによるVirtual Address管理方式をほぼそのまま残して後方互換性を維持しつつ、新たにページ管理方式を追加し、中間アドレス表現としてLinear Addressを定義した
- リニアアドレッシング可能な32 bit (4Gbyte)の広いアドレス空間を持つ
- RISC CPUで主流となったページ型MMUの機能を取り込み、ワークステーション用に開発されたCPUに比べ安価なIA-32 CPUでも、近代的マルチタスクOSが実装可能となり、bsd386, linux等の登場を促すことになる
- マルチタスクOSにおけるメモリ管理としては、4Kbyte/pageの固定長管理で十分であり、page型に比べて実装コストの高いSegment型の可変長ブロックを管理する必要性は低くなっていく
- GDTにLinear Address全域を指定したDescriptorを記述することで、Segment機能を実質的に無視し、page機能のみで仮想メモリ管理を行う事も可能
- 1985年発表



AMD64/Intel 64 （AMD互換なintel）



- x86-64, x64, IA-32e, EM64T等色々な呼称があり、其々仕様が微妙に異なる
- CPUの動作モードにはLegacy modeとLong modeがあり、Legacy modeはIA-32 CPUとの互換モード
- Long modeには更に64bit submode とCompatibility submodeがあり、Compatibility submodeではVirtual Addressが4Gbyteに制限され、Legacy modeのProtect submodeと同一の動作となる
- 以下の説明はLong mode の 64bit submodeについて
  - Global Descriptor, Local Descriptor 共にBase AddressフィールドとLimit Addressフィールドが参照されなくなり、Segment機構は実質的に無効化されている（Virtual Address = Linear Address)
  - リニアアドレッシング可能な64 bit (16 Exa byte)の広いVirtual Address空間を持つ
  - Page Table構成はIA-32の2段構成から最大4段構成に増加
  - 現時点(2018年)では64bit全域をフルデコードしても使い切る事はまず無いと想定し、Virtual Address空間を48bit (256 Tera byte) に制限したCanonical Address Formを使用する。 上位16bit(bits 63-48)は、Effective Address(bits 47-0)のMSB(Most Significant Bit)であるbit47の値をコピーして64bitにアドレス拡張する
  - Physical Address はPTE(Page Table Entry)の制限から最大52 bit(4 Peta Byte)とされており、各CPU実装においてそれ以下のビット数を取る