ハイパーバイザの作り方〜ちゃんと理解する仮想化技術〜 第４回 I/O 仮想化「割り込み編・その１」

割り込みの種類

前回の記事では I/O 仮想化の機能のうち、I/O デバイスへのアクセスをエミュレーションする方法を中心に解説してきました。今回は、割り込み仮想化の話の前提となる x86 アーキテクチャにおける割り込みのしくみを解説します。

CPU が持つ機能として、割り込みというものがあります。これは、現在 CPU 上で実行しているプログラムを停止して、別の処理を実行するための機能です。

CPU より処理速度の遅い周辺機器への I/O を非同期に行うため、周辺機器から CPU へ I/O 終了を通知する機能として実装されました。現在では、より広い用途で用いられています。

外部割り込み、内部割り込み

割り込みは、外部割り込みと内部割り込みの 2 種類に分けられます。外部割り込みとは、ハードウェアから CPU へ「キーボード押された」などのイベント通知を行うのに用いられます。外部割り込みは、さらに**マスク可能な割り込み**と**ソフトウェアからマスク可能な割り込み (NMI)** の 2 種類に分類されます。 マスク可能な割り込みは通常のデバイスからの割り込みに用いられます。また、NMI はハードウェア障害の通知など一部特殊な用途に用いられます。

内部割り込みとは、CPU 内部の要因で発生する割り込みのことです。この内部割り込みは、ソフトウェア割り込みと例外に分類されます。ソフトウェア割り込みとは、割り込みを起こす命令 (INT 命令) により発生する割り込みです。これはシステムコールの実装に用いられます。例外とは、プログラムを実行した結果としてゼロ除算/オーバーフロー/無効な命令の実行/ページフォルトなどが発生したときなどに起きる割り込みです。これは、CPU で実行されるプログラムの制御に用いられます。

ベクタ番号と IDT

これらすべての種類の割り込みに 0-255 のベクタ番号が割り当てられます。例外は要因別に 0-19、NMI は 2 が固定的に割り当てられています。ソフトウェア割り込みは 0-255 のすべてのベクタ番号を使用可能、外部割

り込みは 16-255 を使用可能になっています。

割り込みを使用するには IDT(Interrupt Descriptor Table) と呼ばれる割り込みハンドラのアドレスを格納する最大 255 エントリのテーブル (配列) を作成し、IDT のアドレスを IDTR レジスタに設定する必要があります。割り込み発生時、CPU は IDTR の値から IDT を参照し、指定された割り込みハンドラを実行します。

IDT の各エントリは Gate Descriptor と呼ばれるフォーマットで記述されます。これは単なるメモリアドレスではなく、セグメントの設定や実行権限 (Ring の値)、リアルモード・プロテクトモードの設定などいくつかのパラメータを含みます (図 1)。

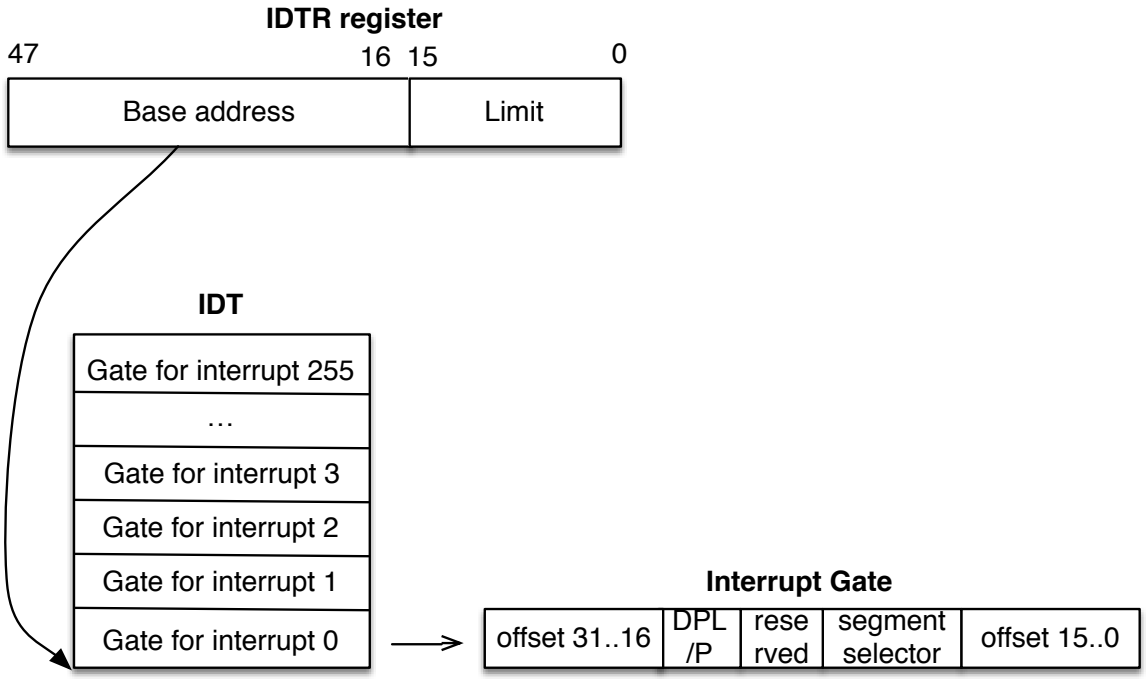

図 1　IDT 、IDTR、Gate descriptor の関係

IDT に用いられる Gate Descriptor には Task gate descriptor/Interrupt gate descriptor/Trap gate descriptor の 3 種類があります。しかし、Task gate descriptor はハードウェアによるマルチタスク機能を呼び出すためのもので、現代の OS では使われていません。代わりに Interrupt gate descriptor と Trap gate descriptor が用いられます。

この 2 つの違いは、Interrupt gate descriptor が EFLAGS レジスタの IF フラグ (割り込みフラグ) をクリアするのに対して、Trap gate descriptor は IF フ ラグをクリアしない (割り込みハンドラ内で割り込みを禁止しない) というものです。

IDTR レジスタのフォーマットは、IDT の先頭アドレスと Limit 値 (テーブルのサイズ) の組み合わせになっており、255 未満のテーブルサイズに対応できるようになっています。

割り込みのマスク・アンマスク

EFLAGS レジスタ (Intel 64 では RFLAGS レジスタ) の IF フラグをクリアすると割り込みを禁止、IF フラグをセットすると割り込みを有効にすることができます。プログラムから IF フラグをクリア/セットするには、CLI 命令/STI 命令を使用します。

また、前述のとおり、Interrupt Gate Descriptor を使用して割り込みをハンドルする場合は CPU によって IF フラグがクリアされるため、割り込みは自動的に禁止されます。このとき、割り込みハンドラを終了して割り込み前に実行していたプログラムへ制御を戻す IRET 命令を実行すると、割り込み前の EFLAGS レジスタの値がリストアされ、再び割り込み可能になります。

外部割り込みと割り込みコントローラ

オリジナルの PC アーキテクチャでは、外部割り込みを制御し、CPU へ伝える役割を受け持つ割り込みコントローラとして PIC (i8259) がありました。これは、マザーボード上に存在し、CPU の割り込みラインに接続されていました。Pentium 以降の x86 アーキテクチャでは、外部割り込みの管理は APIC (Advanced Programmable Interrupt Controller) と呼ばれる新しい割り込みコントローラへ移行され、PIC は互換性のためのみに存在するようになりました。APIC は CPU ごとに存在し、CPU に内蔵さ れている Local APIC と、ICH (Southbridge) に内蔵されている I/O APIC から構成されています (図 2)。

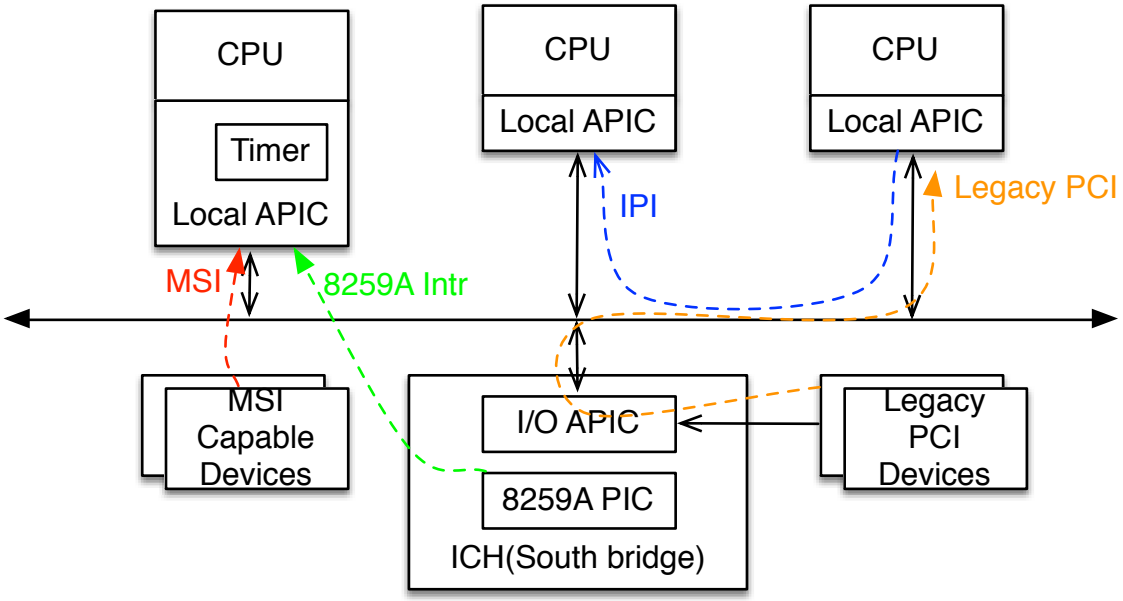

図 2　Local APIC と I/O APIC、外部デバイスの関係

Local APIC は、ローカル割り込みのベクタ番号設定、割り込みベクタ番号通知、EOI (割り込み終了) 通知な

ど、一般的な割り込みコントローラの役割を受け持ちます。また、タイマー/温度センサ/パフォーマンスモニタリングカウンタなどのデバイス、 IPI (プロセッサ間割り込み) の送受信機能を内蔵しています。

一方、I/O APIC は外部デバイスから割り込みを受け取り、I/O APIC 上の設定に基いて割り込み配信先の Local APIC を選び割り込みをリダイレクトする機能を受け持ちます。

このような構成を取ることによって、SMP 環境下で複数の CPU で外部割り込みを処理できるようになっています。

また、ある CPU の Local APIC から別の CPU の Local APIC へ IPI (Inter-Processor Interrupt:プロ セッサ間割り込み) を送ることもできます。

Local APIC

表 1 に割り込みの処理で利用される Local APIC の 主なレジスタを示します。

レジスタ名	内容
IRR（Interrupt Request Register）	割り込み要求レジスタ（Read-only）：未処理の割り込みを管理するレジスタ。 Local APIC に外部割り込みが着信する度にベクタ番号に対応するビットがセットされる。
ISR（In-Service Register）	インサービスレジスタ（Read-only）：次に発行される割り込みの候補を管理するレジスタ。 EOI レジスタへ書き込まれると Local APIC によって IRR から最高優先度のビットがコピーされる。
EOI（End Of Interrupt）	割り込み終了レジスタ（Write-only）：割り込み処理の終了を Local APIC に通知する為のレジスタ。
TPR（Task Priority Register）	タスク優先度レジスタ（Read/write）：外部割り込みに対する実行中タスクの優先度を設定するレジスタ。 TPR に書き込まれた優先度より低い優先度の割り込みはマスクされる。
PPR（Processor Priority Register）	プロセッサ優先度レジスタ（Read-only）：実際の割り込み着信時にマスクを行うか否かの判定に使われるレジスタで、TPR と ISR に連動して更新される。
Local APIC ID Register	LAPIC ID レジスタ（Read/Write）：システム全体で CPU を一意に特定する為の ID である LAPIC ID を格納しているレジスタ。 LAPIC ID は I/O APIC から外部割り込みを転送する時や IPI を送るときなどに使用される。

レジスタ名	内容
ICR（Interrupt Command Register）	割り込みコマンドレジスタ（Read/write）：IPI（プロセッサ間割り込み）を送信する為のレジスタ。
LDR（Logical Destination Register）	Logical APIC ID を指定
DFR（Destination Format Register）	Logical Destination Mode のモデルを指定（Flat model/Cluster model）

表 1: Local APIC の主なレジスタ

割り込み着信時の Local APIC および CPU の挙動を簡単にまとめると、次のよう流れになります。

1. Local APIC が割り込みを受信したら、IRR に対応するベクタ番号のビットをセットする。CPU が割り込みをブロックしている場合はここで処理は終わり
2. IRR にセットされた最高優先度のビットをクリア、同じビットを ISR にセット、同じ優先度の割り込みを CPU へ発行する
3. CPU で割り込みハンドラが実行される
4. CPU で実行された割り込みハンドラが EOI レジスタに書き込み、割り込み処理の終了を伝える
5. EOI レジスタへの書き込みを受け取ると、ISR にセットされた最高優先度のビットをクリア。まだ ISR にビットが残っていたら 2 から繰り返し

ただし、タスク優先度/プロセッサ優先度という機能があり、これによって割り込みに対する実行中タスクの優先度が制御でき、タスクの優先度が受信した割り込みより高い場合、その割り込みはマスクされます。

TPR ・ PPR レジスタの値は優先度クラス (4-7bit)・サブ優先度クラス (0-3bit) の 2 つの値からなり、優先度クラスだけが割り込みのマスクに使用されます。先度サブクラスは後述の「Lowest priority Mode」で使われますが、割り込みのマスクには使用されません。

PPR の更新は TPR に新しい値が書き込まれたときと CPU に割り込みを発行するとき (前述の割り込み着信時の手順 2 の段階) に行われます。値は次のように設定されます。

```
/* [7:4] は優先度クラス */
/* [3:0] は優先度サブクラス */
if TPR[7:4] >= ISR[7:4]
  PPR = TPR
else
  PPR[7:4] = ISR[7:4]
  PPR[3:0] = 0
```

また、Local APIC に割り込みが着信した段階 (前述の割り込み着信時の手順 1 の段階) で、IRR にセットされた最高優先度のビットと PPR の優先度クラス (4-7bit) が比較されます。

PPR の優先度クラス (4-7bit) のほうが高かった場合は割り込みがマスクされます。

I/O APIC

I/O APIC には外部デバイスからの割り込み線が接続され、24 本の割り込みをサポートしています。各割 り込みの割り込み先は I/O APIC 上の Redirection Table に設定され、システムバスを通じて CPU の Local APIC へ転送されます。Redirection Table の各エントリのフィールドの詳細を表 2 に示します。

ビットポジション	フィールド名	内容
63:56:00	Destination	宛先 Local APIC の指定
16	Mask	割り込みマスク
15	Trigger Mode	エッジトリガ／レベルセンシティブ
14	Remote IRR	レベルセンシティブモードで割り込み送信中／ EOI 着信済み
13	Interrupt Pin Polarity	レベルセンシティブモードで high/low のどちらを 割り込みリクエストとするか
12	Delivery Status	割り込みペンディング中かどうか
11	Destination Mode	Physical Mode ／ Logical Mode
10:08	Delivery Mode	Fixed ／ Lowest Priority など
7:00	Vector	ベクタ番号

表 2: Local APIC の主なレジスタ

Redirection Table Entry では Destination Mode を指定する必要があります。Destination Mode に Physical Destination Mode を指定した場合、 Destination フィールドに Local APIC ID を指定することによ り、宛先 CPU を一意に特定します。たとえば、Local APIC ID: 3 の CPU へ割り込みを配信したいときは、Destination フィールドに 3 を指定します。

Logical Destination Mode を指定した場合、Destination フィールドで複数の宛先 CPU 群をビットマスク指定します[*1]。宛先 CPU の ID は Local APIC ID ではなく、LDR の Logical APIC ID が使用されます。たとえば、LDR の値が 00000001b の CPU と 00000010b の CPU に割り込みを配信したい場合、Destination フィールドに 00000011b を指定します。

Logical Destination Mode により複数の CPU が割り込み先に指定された時の挙動については、 Delivery Mode で設定します。Fixed Mode では、 Destination に指定されたすべての CPU に割り込みます。Lowest

[*1] これは正確には Logical Flat Model というモードで、他に Flat Cluster Model、Hierarchical Cluster Model などがありますが、通常使われません。説明は割愛します。

priority Mode では、Destination のうち Local APIC の TPR が最も小さい CPU へ割り込みます。TPR の値が最も小さい CPU が複数存在する場合は、ラウンドロビンで割り込みが分散されます[*2]。いずれの配信モードの場合でも、Vector フィールドで設定されたベクタ番号で Local APIC へ割り込みがかかります。

APIC ID のアドレス幅は 8bit ですので、Logical Destination Mode では最大 8CPU しかサポートできません。このため、Nehalem 世代より x2APIC と呼ばれる拡張版 APIC が導入され、アドレス幅は 8bit から 32bit へ拡張されました (“Intel64 Architecture X2APIC Specification”) 。

MSI・MSI-X 割り込み

I/O APIC を経由する PCI デバイスの割り込みは、各 PCI デバイスからの物理的な割り込み線が I/O APIC へ接続され、この割り込み線を通じて割り込みが送信されていました。

この構成では割り込みの数と IRQ の割り当てが物理配線に依存するため、限られた IRQ を複数のデバイスで共有するようなしくみになっていました。また、1 つのデバイスで複数の割り込みを持つことはできませんでした。この制限を解消するため、物理配線を用いず PCI バス経由のメッセージとして割り込みを送る Message Signalled Interrupt (MSI)・Extended Message Signalled Interrupt (MSI-X) が導入されています。PCI ではオプション機能として提供されていますが、PCI express では必須とされています。

MSI では各デバイスごとに 32 個、 MSI-X では 2048 個の割り込みをサポートします。従来と異なり、デバイス間の割り込みは共有されません。MSI ・ MSI-X の割り込みは IO-APIC を経由せず、直接 Local APIC へ配送されます (図 2 参照)。このとき、宛先 CPU の設定は各 PCI デバイスのコンフィギュレーションスペースに設定されます。詳細は割愛しますが、コンフィギュレーションスペース内の割り込みの設定フィールドでは、表 2 の Redirection Table Entry に近い内容が割り込みごとに設定できます (“最近の PC アーキテクチャにおける割り込みルーティングの仕組み”) 。

まとめ

いかがでしたでしょうか。今回は x86 アーキテクチャにおける割り込みのしくみを解説してきました。次回は、今回解説した割り込みのしくみをどう仮想化するかについて解説します。

ライセンス

[*2] Windows ではプロセスごとに TPR の値を制御し、プロセスの優先度に多じて割り込み量を変化させています。一方、Linux では TPR の値は変えずに Lowest Priority を用い、複数の CPU へ割り込みを公平に分散させています。

参考文献

“Intel64 Architecture X2APIC Specification.” http://www.intel.com/content/dam/doc/specification-update/64-architect

“最近の PC アーキテクチャにおける割り込みルーティングの仕組み.” http://syuu1228.github.io/howto_implement_hyperv