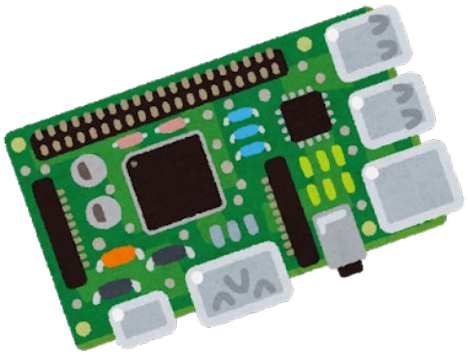
■#05 CPU の高速化と問題点

コンピュータは、初期の頃とても高価であったため、利用できるリソースが限られており。様々な工夫をしながら利用していた。また性能向上の仕組みに取り組み、現在のコンピュータの基礎的な構造をなしています。



実は最近話にあるセキュリティ・ホールとなっているのが、このあたりの技術です。振り返りながら、最新の情報をたぐってみましょう。

1. 発表されたプロセッサの問題点

2018 年 1 月 4 日に、Google の Project Zero と Cyberus Technology と グラーツ大学の研究者らによって別々に発見され、Spectre という別の脆弱性とともに、発表されました。

Intel が「この脆弱性は Intel だけのものではなく、他のプロセッサにも存在する」と発表し、大きく話題になりました。
これは業界にとって相当に深刻な問題となっています、そこで今回は、プロセッサの脆弱性の「仕組み」について図解します。

2. プロセッサの基礎知識

この脆弱性には、「Spectre」と呼ばれる「Variant(亜種) 1」「Variant 2」の脆弱性と、「Meltdown」と呼ばれる「Variant 3」の脆弱性の 3 種類があります。

このうち説明のしやすい Meltdown について説明しますが、その前に、脆弱性が起こる仕組みを理解するための、プロセッサについての基礎知識を振り返りかえってみます。

3. CPU の特権モード

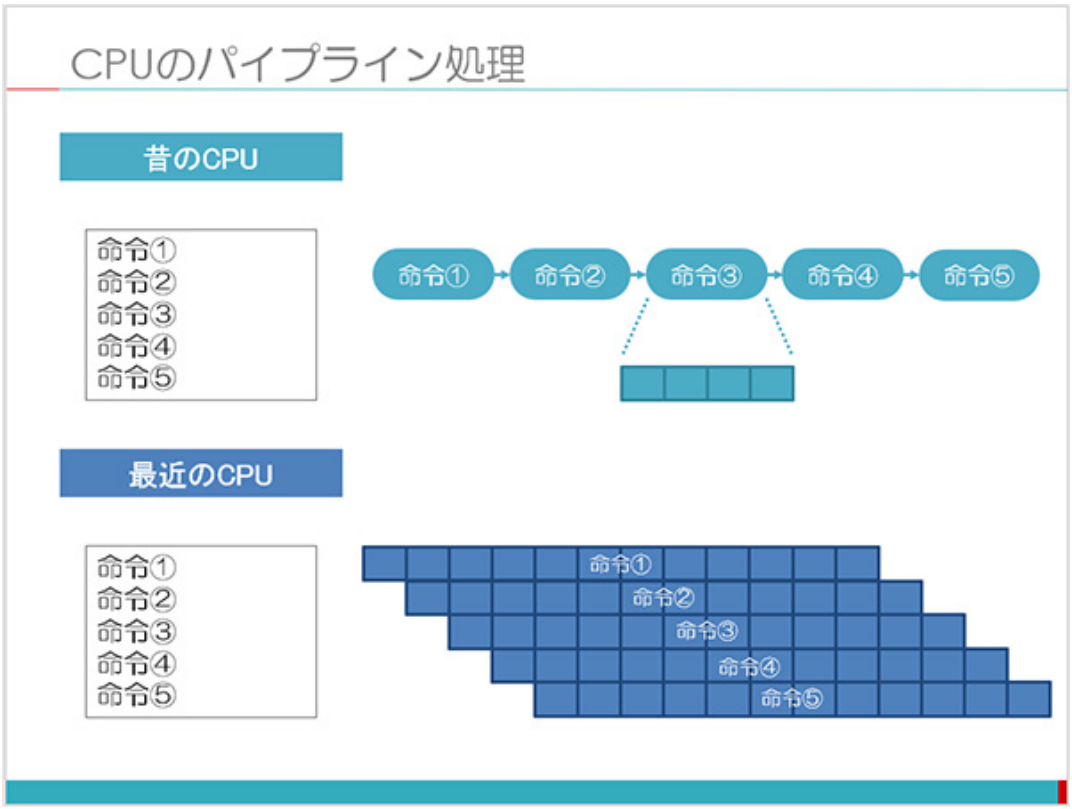
現代のプロセッサには「特権モード」というのがあり、プロセスに優先度を付けています。OS やデバイスドライバなど、優先的に実行させたいプロセスに与えられるのが最高特権である「カーネルモード」、それよりも優先度が落ちるのが「ユーザーモード」です。

カーネルモードがプロテクトしているメモリエリアには、通常、ユーザーモードのプロセスはアクセスできませんが、今回の脆弱性はそれを可能にしてしまうというものです。

4. パイプライン処理

現代のプロセッサでは、命令の処理を同時並行して行う「パイプライン処理」が主流になっています。

昔のプロセッサは、命令を 1 つずつ読み込んで実行し、処理が終わると次の命令にとりかかる、というやり方をしていました。



個々の命令は、プロセッサの中でさらに細分化されます。例えば、（1）命令を解読（デコード）する（2）レジスタを参照する（3）足し算をする といった形です。

そうすると、命令をデコードした後で他の処理に移行した場合、次の命令が来るまでデコードするための回路は遊んでしまうことになります。プロセッサはさまざまな回路の集合体ですが、“ある時点でその一部しか稼働していない”ということになります。

これはもったいない、ということで、それぞれの命令を整理して細分化し、前の命令の処理が完了しないうちに次の命令を、空いた部分の処理を利用して実行開始するようになりました。これがパイプライン処理です。最近では数十のステージに細分化されています。

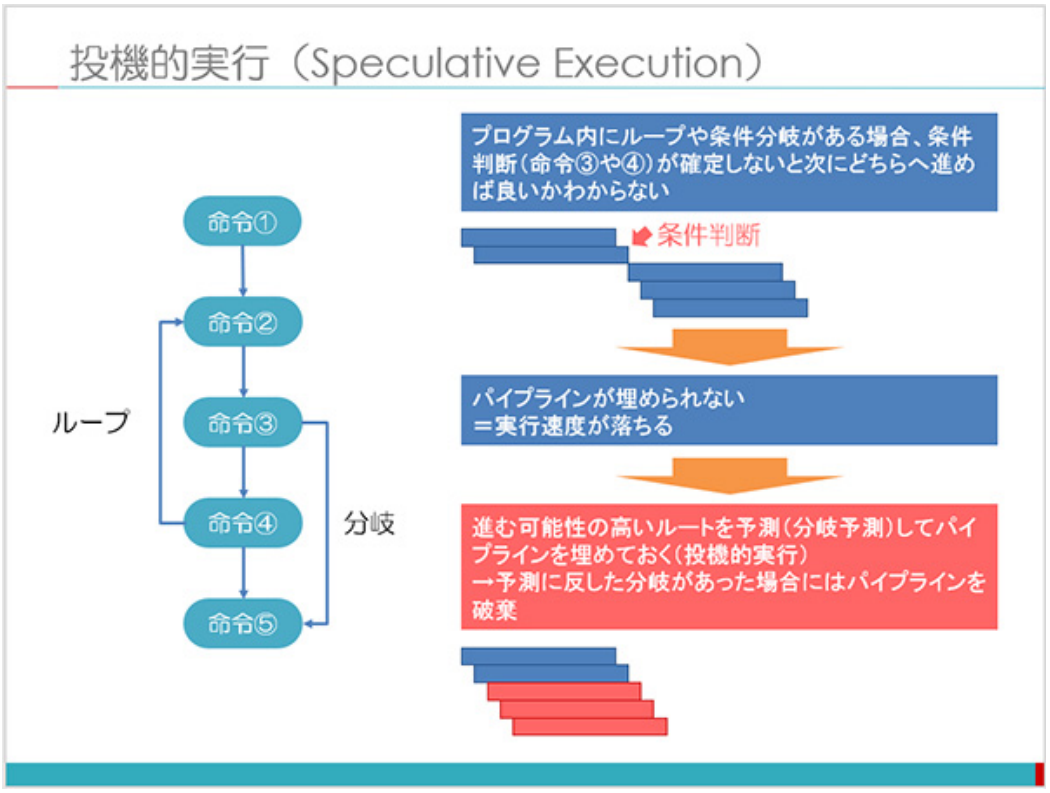
このような処理を効率よく実行するためには、命令長が固定であることが重要になります。

昔の CISC（シスク）という CPU アーキテクチャでは、高性能化のために各命令長が異なり、命令によって実行時間が大幅に異なっていました。そのため、パイプラインを導入することが難しく、回路も複雑化していました。

そこで、命令を比較的低機能なものにして、命令長を固定にし、パイプラインを利用して高クロックで命令を処理し全体のパフォーマンスを上げる、RISC（リスク）という CPU アーキテクチャが一般的になっています。

5. アウトオブオーダー実行

プログラムには、ループや条件が付きものですが、分岐するかどうかはその命令を終えて、条件が確定しないと判断できません。



そうすると、この場合も前の命令が終わるまで次の命令を開始できないことになります。特に科学技術計算では、ループを多用しますから、処理速度に大きな影響を与えます。

そこで、考え出されたのが「投機的実行」です。条件が確定する前にとにかく次の命令を実行してしまい、もし駄目だったらそれまでの結果は破棄して正しい命令にとりかかるということです。

“イチかバチか”ということで「投機的」なのですね。もちろん、失敗が続くと処理速度が遅くなってしまいますから、さまざまな手段を使って予測精度をあげる工夫がされています。「分岐予測」などもその１つです。

またこのようにパイプライン処理にかかる不具合をパイプラインハザードといい、他にもデータハザード、制御ハザード、構造ハザードがあります。

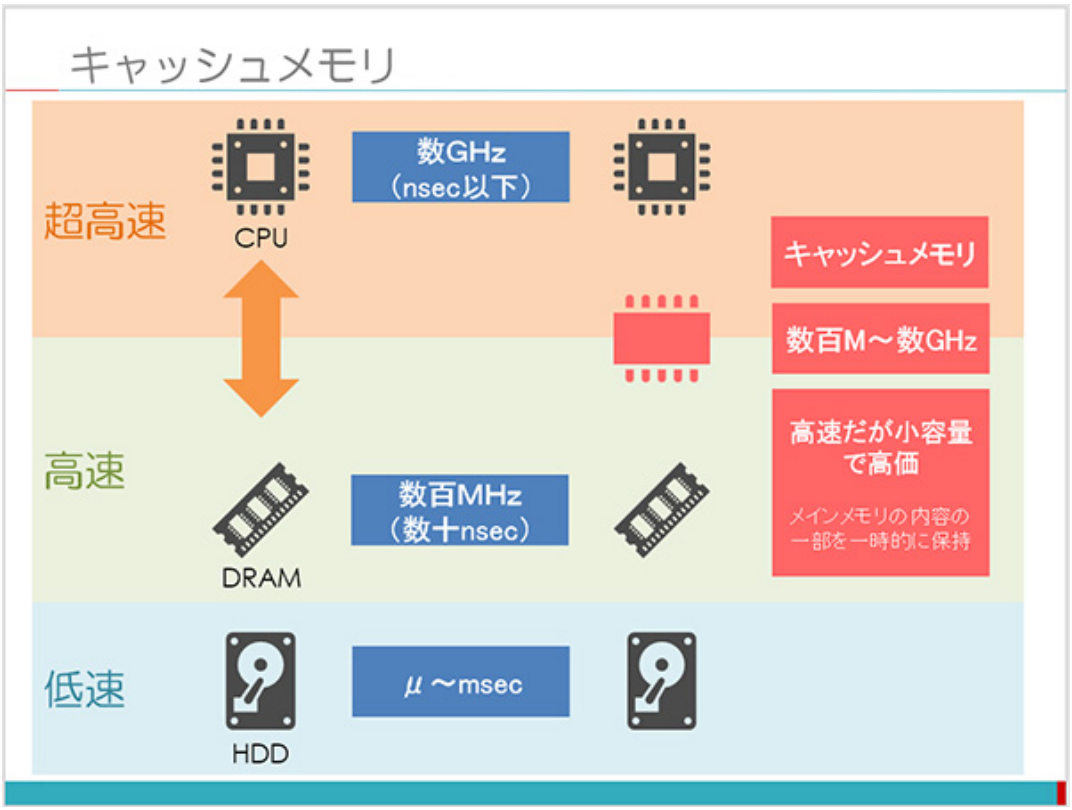
6. キャッシュメモリ

パイプライン処理や半導体技術の進化によって、プロセッサ内部での処理は高速化していきますが、メインメモリである DRAM へのアクセス速度は上がっていません。そのため、メモリからのデータを待つためにプロセッサのパイプラインが止ってしまうことになります。

そこで、メインメモリよりも高速なメモリを間に配置し、データアクセスを高速化しようとするのが「キャッシュメモリ」です。

プロセッサがメインメモリからデータを読み込むとき、ひとまとまり（数十～数百バイト）のデータがキャッシュに読み込まれます。データはまとまって置かれていることが多いため、プロセッサが次のデータを読もうとしたときに、キャッシュにデータがある可能性が高いのです。

キャッシュに目的のデータがあれば、メインメモリまで読みに行かなくても、そのデータを読み込むことができ、処理の高速化につながります。（キャッシュにもいろいろな種類がありますが、これはリードスルーキャッシュです。）

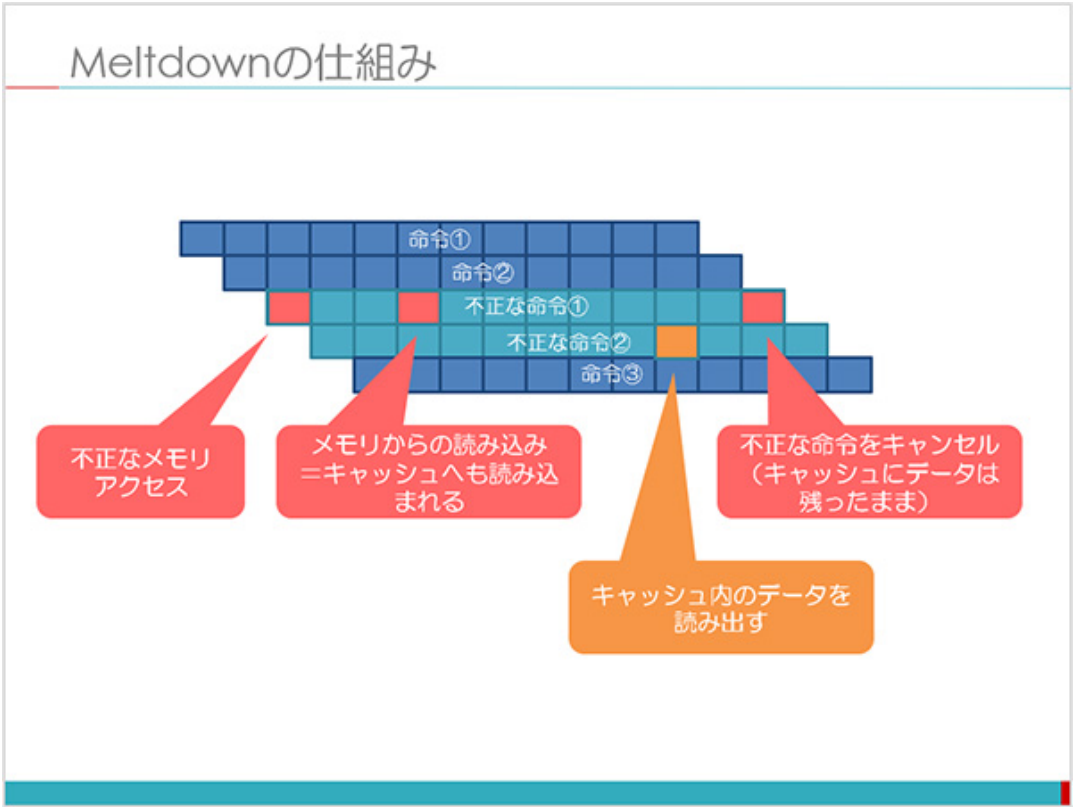


全てのメモリを高速化すればよさそうなものですが、高速なメモリはコストがかかるのです。

現代のプロセッサは、キャッシュメモリをプロセッサ内部に持っており、パイプラインと同期して動作させることができます。ここで大事なのが、当面、プロセッサが要求していないデータもキャッシュに読み込まれることと、そのデータが使われなくても特に操作をしない限りデータはキャッシュに残ることです。ただ通常はデータの読み込みが続いてキャッシュが足りなくなれば、古いデータは上書きされることになります。

7. Meltdown の仕組み

これで Meltdown の説明ができる準備が整いました。ある不正なプログラムが、不正な命令、例えば本来ユーザープロセスからアクセス不可能なメモリエリアからのデータ読み出しを発行したとしましょう。



この不正な命令自身は、プロセッサが命令を実行する過程で不正なメモリアクセスであることが認識され、命令の実行はキャンセルされます。

しかし、不正な命令であることが認識されないステージでメモリアクセスが行われた場合、メモリからキャッシュへのデータ読み込みは実行されてしまいます。命令がキャンセルされても、キャッシュ上のデータは残ります。

ここで攻撃者が引き続きメモリをアクセスする命令を絶妙なタイミングで発行した場合、キャッシュに残った（本来アクセスしてはいけない）データを読み取ることが（理論的には）可能になります。

しかし、この脆弱性を悪用した実際の攻撃は現時点で確認されていません。いくつかの命令を絶妙なタイミングで実行させなければならないなど、技術的難易度は相当高いといわれていますが、ハードウェアのアーキテクチャレベルで脆弱性が見つかったこと自体が、業界に大きな衝撃を与えているわけです。

〇チェックポイント・キーワード

・CPU アーキテクチャ
・RISC/CISC
・アウトオブ・オーダー
・パイプライン
・キャッシュ、レジスタ
・セキュリティ