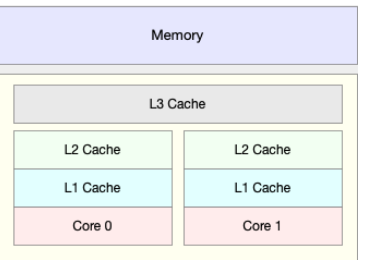
Meltdown原理介绍

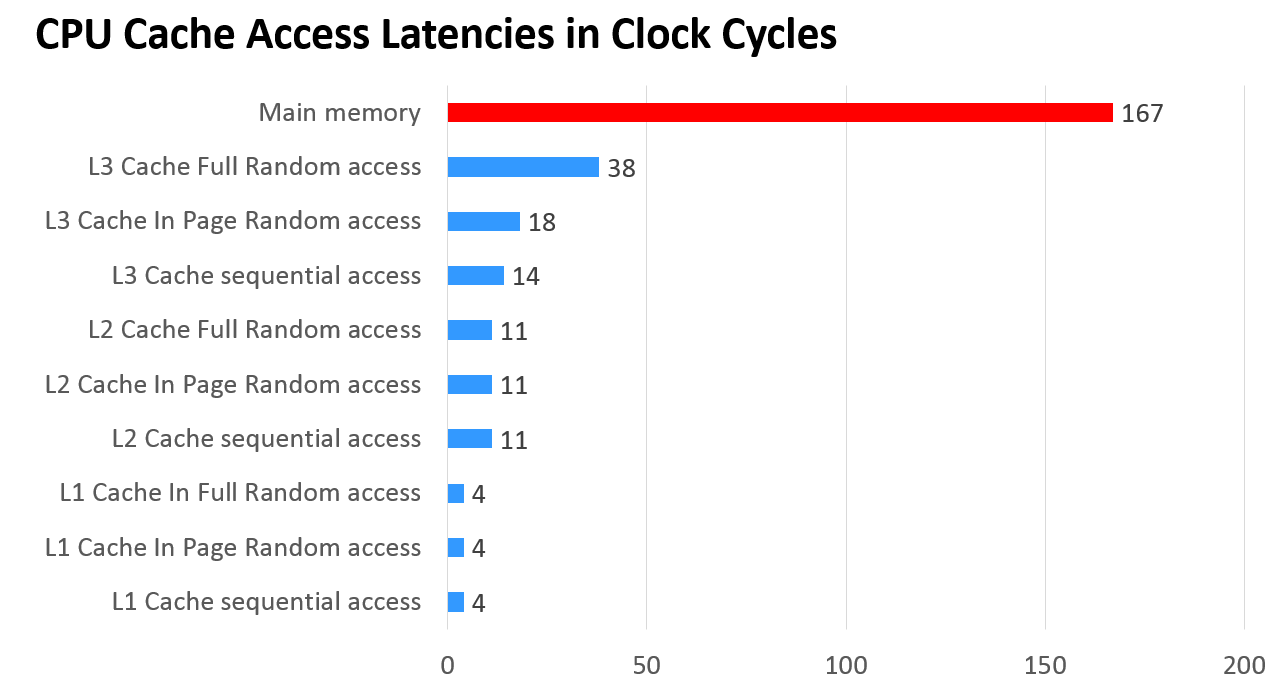
在深入分析Meltdown之前，我们需要了解一些背景知识。它包括CPU Cache，CPU指令执行，操作系统地址空间隔离的设计。

**1 CPU Cache**

现代处理器执行指令的瓶颈已经不在CPU端，而是在内存访问端。因为CPU的处理速度要远远大于物理内存的访问速度，所以为了减轻CPU等待数据的时间，在现代处理器设计中都设置了多级的cache单元。



可以从下图看出CPU对cache和内存的访问时钟周期，对于CPU内部的寄存器，一般用于存储程序中局部变量，其访问时间大概为一个时钟周期，可想而知对于一个时钟为4GHz的CPU其一个时钟周期有多块。此外，下图没有说明磁盘的访问时间，一般而言，CPU访问磁盘的时间大约在秒级左右。

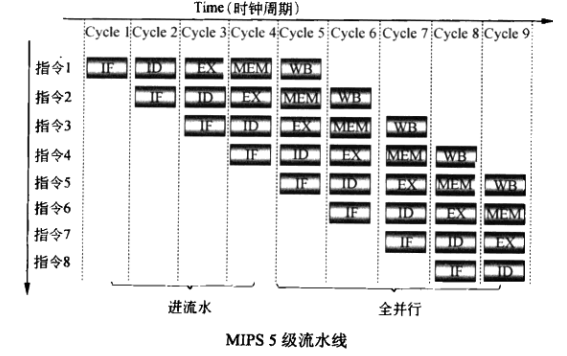


**2 Cache攻击**

我们已经知道同一个CPU上的Core共享L2 cache和L3 Cache, 如果内存已经被缓存到CPU cache里, 那么同一个CPU的Core就会用较短的时间取到内存里的内容, 否则取内存的时间就会较长。两者的时间差异非常明显(大约有300个CPU时钟周期), 因此攻击者可以利用这个时间差异来进行攻击。

**3 指令执行**

**经典指令流水过程：取指---译码--执行--取数---写回**

****

现代处理器在设计上都采用了超标量体系结构（Superscalar Architecture）和乱序执行（Out-of-Order）技术，极大地提高了处理器计算能力。超标量技术能够在一个时钟周期内执行多个指令，实现指令级的并行，有效提高了ILP（InstructionLevel Parallelism）指令级的并行效率，同时也增加了整个cache和memory层次结构的实现难度。

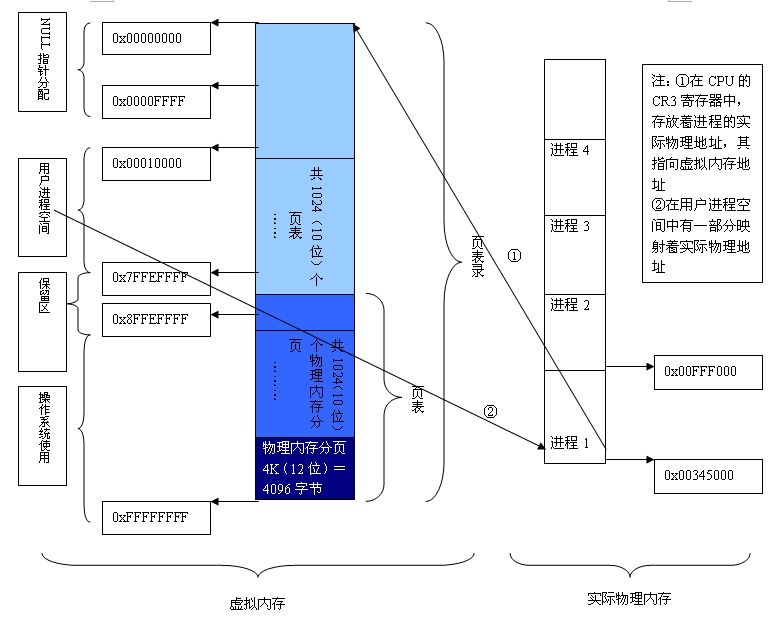
**4 乱序执行(out-of-order execution)**

乱序执行（out-of-order execution）是指CPU采用了允许将多条指令不按程序规定的顺序分开发送给各相应电路单元处理的技术。比方Core乱序执行引擎说程序某一段有7条指令，此时CPU将根据各单元电路的空闲状态和各指令能否提前执行的具体情况分析后，将能提前执行的指令立即发送给相应电路执行。

在各单元不按规定顺序执行完指令后还必须由相应电路再将运算结果重新按原来程序指定的指令顺序排列后才能返回程序。这种将各条指令不按顺序拆散后执行的运行方式就叫乱序执行（也有叫错序执行）技术。

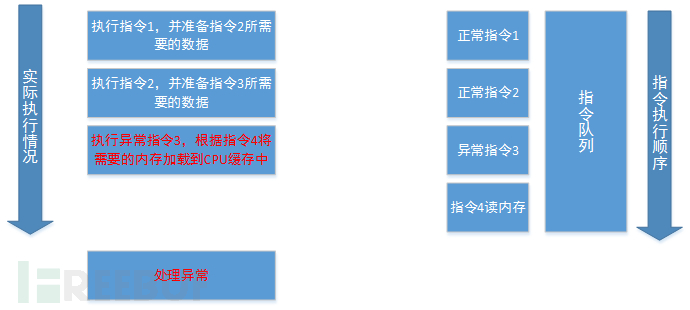
通常处理器实现了一个统一的保留站（reservation station），它允许处理器把已经执行的指令的结果保存到这里，然后在最后指令提交的时候会去做寄存器重命名来保证指令顺序的正确性。  
 经典的X86处理器中的“整数重命名”和“浮点重命名”部件（英文叫做reorder buffer，简称ROB），它会负责寄存器的分配、寄存器重命名以及指令丢弃（retiring）等作用。  
x86的指令从L1 指令cache中预取之后，进入前端处理部分（Front-end），这里会做指令的分支预测和指令编码等工作，这里是顺序执行的（in-order）。指令译码的时候会把x86指令变成众多的微指令（uOPs），这些微指令会按照顺序发送到执行引擎（Execution Engine）。执行引擎这边开始乱序执行了。这些指令首先会进入到重命名缓存（ROB）里，然后ROB部件会把这些指令经由调度器单元发生到各个执行单元（Execution Unit，简称EU）里。假设有一条指令需要访问内存，这个EU单元就停止等待了，但是后面的指令不需要停顿下来等这条指令，因为ROB会把后面的指令发送给空闲的EU单元，这样就实现了乱序执行。

**5 地址空间**

****

虽然两者是在同一张页表里，但是他们对应的权限不一样，内核空间部份标记为仅在特权层可以访问，而用户空间部份在特权层与非特权层都可以访问。这样就完美地把用户空间和内核空间隔离开来：当进程跑在用户空间时只能访问用户空间的地址映射，而陷入到内核后就即可以访问内核空间也可以访问用户空间。  
 对应地，页表中的用户空间映射部份只包含本机程可以访问的物理内存映射，而任意的物理内存都有可能会被映射到内核空间部分

**6 cache攻击原理**

****

对于具有预测执行能力的新型处理器，在实际CPU执行过程中，指令4所需的内存加载环节不依赖于指令3是否能够正常执行，而且从内存到缓存加载这个环节不会验证访问的内存是否合法有效。即使指令3出现异常，指令4无法执行，但指令4所需的内存数据已加载到CPU缓存中，这一结果导致指令4即使加载的是无权限访问的内存数据，该内存数据也会加载到CPU缓存中，因为CPU是在缓存到寄存器这个环节才去检测地址是否合法，而CPU分支预测仅仅是完成内存到CPU缓存的加载，实际指令4并没有被真正的执行，所以他的非法访问是不会触发异常的。