CPU reset

作者： 本文由 freevanx（freevanx@gmail.com）起草，

	Version 1.1 修订

关于这个 topic，已经有很多资料描述。本文希望能够总结有关这个 topic 相关的资料，整理一个比较清晰文档来描述这个问题。本文只是 EFI 一系列文档的一部分，其实就本文的内容来说大部分是与 EFI or legacy 无关的。

CPU reset 之 HW

在 platform 发出 PLTRST#之前, HW 需要正确的按照 power sequence 的要求上电，本文不会对 power sequence 做详细解释，这部分需要 HW 的工作来保证，不同的 platform，其 power sequence 的要求也不同。

在 CPU reset 和 BIST 之后，CPU 将会去读取第一条指令。对于传统的 CPU + 北桥 + 南桥类型的 platform 来说， CPU 的 request 通过 FSB 到达北桥，北桥将这个 request 透过 ESI 送到南桥。而对于最新的 chipset 来说，北桥和 CPU 封装在一颗 Chip 里面，所以会看到这个 request 通过 DMI/QPI 被送到南桥。

Request 到达南桥后，南桥根据配置决定将 request route 到 SPI 或者 LPC。而 LPC 又可分为 FWH 或者直接 LPC request 到 EC。在旧的 chipset 上，很多都是支持 Firmware Hub 接口的 BIOS ROM，Firmware Hub（FWH）接口是 LPC 的一种变形，通过扩展，使在同一个 bus 上支持多颗 Flash ROM，FWH 接口的 Flash 一般使用 PLCC 封装，体积巨大，占用了很大的 PCB 空间，加之速度不如新的 SPI Flash，所以现在基本被 SPI Flash 取代了。SPI Flash 基于 SPI Bus，一种简单的串行 4 线总线。SPI Flash 一般采用 SOP 封装，体积小，且一般只有 8 pin，相对于 FWH Flash，有体积和速度上的优势。在南桥上，一般都会有一个 strap pin, 用来选择将 request route 到 SPI 或者 LPC。

通常再确认 CPU 有无开始抓 BIOS Code 的时候，会通过测量 SPI 或者 LPC 的信号来确定。对于 SPI Flash，在测量到 SI 和 SO 都有信号时，才可以认定 CPU 的确有抓 code，如果有经验的工程师，可以通过测量 CLK SI SO 3 个信号，follow SPI spec，解读 CPU 到底读写了那些 data。而对于 FWH flash 或者 flash 挂在 EC 下的 case 来说，首先可以通过测量 LPC Bus 的 LFrame#信号来做初步判定，根据 LPC spec，LPC 在每次读写时，都会拉低 LFrame# 作为开始的信号。当然如果手上有 Flash ROM 的 Emulator 的话，例如 Dediprog 的 EM100 SPI flash Emulator, 可以使用 Emulator 的 capture bus data 的 function 来判定，如果 capture log 有 bus data 的读写，自然可以判定 CPU 是有读 code 的。

Note：EM100 是 Dediprog 公司的产品，有关解释权利属 Dediprog 所有。

CPU reset 之地址映射

在 CPU 读到 BIOS Code 之前，一切工作都是由 HW 完成，其中南桥做了一些很重要的幕后工作。Route Flash ROM 读取的 request 是其中重要的一项，上文中提到南桥会把收到的来自 CPU 的 request 转发到 SPI bus 或者 LPC，这里仅仅简单的提到了第一条指令，然而，事实上，并不是所有地址空间的 request 都会受到特别关照，能够被转发到 SPI Bus 或者 LPC bus。

这要从 PCI Bridge 的 decode 说起，PCI 有 3 中 decode 方式，而 DMI bridge/ESI bridge 默认是 subtractive decode，在这种 decode 方式下，DMI/ESI bridge 会把所有别人不想要或者不需要的 request 转发给南桥。而南桥自己有一张特殊的 address mapping table，类似于关系表之类的东西，里面记录了类似南桥的亲戚朋友，见过面的，路人甲，路人乙之类的复杂关系。如果 request 的地址不在南桥的关系表中，也没有设备主动要这个 request，那么最终南桥都会回一个 0xFF 给 CPU。而能够被 route 到 SPI 或者 LPC 的 request，也是属于南桥特别关照的亲戚朋友的一员。事实上，南桥存在两个这样的区域，第一个是从地址空间 4G 向下，大小从 4MB 到 16MB 不等的一个区域，在 Mobile platform 上，由于相对简单，BIOS code 也相对较小，所以一般在 4MB 左右，而在 Server Platform 上，则较大，可能为 16MB 或者更大。以 4MB 为例，地址空间从 FFFC00000h ~ FFFFFFFFh。称之为 Range 4G。已经了解 CPU reset 的朋友显然知道，CPU reset 后会从 FFFFFFF0h 抓第一条指令，很显然，这第一条指令也正好处于南桥的亲戚 4G 的范围内，能享受南桥的 special service（Note：插一句题外话，Sevice 这个词，实在是太邪恶了）。 到这里，大家应该知道，CPU 抓的第一条指令能最终被 Flash ROM 收到，这并不是偶然的，因为这个 request 和南桥是有微妙的关系的。当 CPU 发出抓 code 的请求后，接下来 DMI/ESI 收到这个请求，转发给南桥，南桥发现这个请求的地址在自己的关系表中，就根据设置将其转发给 SPI 或者 LPC。 接下来，第二个受到南桥关照的地址空间，一般是从 1MB 向下 128KB 的范围，即 E Segment 和 F Segment，从 E0000~FFFFF， 称之为 Legacy Range，也就是说，不仅 FFFC00000h ~ FFFFFFFFh 之间的 request 能够被 route 到 SPI/LPC，E0000~ FFFFF 之间的 request 也同样会被 route 到 SPI/LPC。关于 Legacy Range 的来历，那应该是 long long ago 的事情，也许 8086/80186 的年代第一条指令就是从 FFFF0 读取吧，本人没有去查证过，因为不是写历史的。那至于为什么时至今日，Legacy Range 还被 chipset 保留，甚至没有办法不 support，可以查考南桥写的自传《我和 Legacy Range 不得不说的那些事》，（Note：开玩笑的）详情会在下 3 回分解。

所以因为这些事情，有文档说，CPU 第一条指令会从 FFFFFFF0h 抓，而有文档说是从 FFFF0 抓，那到底从那里抓呢？一切以 Intel 的说法为准，Intel 说从 FFFFFFF0h 抓，那就是从 FFFFFFF0h 抓。

另外，可能有一些文档中说“Cpu reset 时，会从内存 xxxx 的位置读取第一条指令”，那么，反问一个问题，CPU reset 的时候，内存都没有初始化，哪来的内存可给你用？这其实存在概念上的误解，CPU 从来不会直接从内存中读数据，CPU 的读写，都是针对自己的地址空间而言。 那么，x86 平台上内存如何工作呢？想象一下网络的概念，bus 相当于网络，CPU 是其中一台主机，内存控制器是另一台主机，而南北桥上所有的 PCI 设备都是一台独立的主机，CPU 抓取可执行的代码或数据，相当于 CPU 主机向网络某个地址发送了一个读写请求，具体读到的东西是什么，取决于这个地址上的设备，如果这个地址上是一个内存控制器主机，那么请求就可能被送到内存中， 如果这个地址上是一个 SPI bridge，那么请求就被送到 SPI rom 上， 如果是一个 LPC bridge， 请求就会被送到 LPC。假设一个 CPU，其地址空间大小是 4GB，这个系统拥有一根 1GB 的内存。内存初始化过后，其地址被映射到 0~3FFFFFFFh 的地址空间，假设 CPU 要访问一个地址 10000h，CPU 的 request 首先针对自己的地址空间，然后 CPU 的硬件会将这个请求转化为一个 bus 的请求，比如 FSB or PCI，然后内存控制器从 bus 上收到这样一个请求后，会确认这个请求是否在内存的地址空间，如

果是在内存映射的空间内，则将其重定向到内存中，然后就由内存回复这个请求。如果 CPU 访问一个地址 8FFFFFFFh，当这个请求到达 bus 被内存控制器收到后，经过比较，这个地址不在内存映射的范围内，所以内存控制器将不处理这个请求，然后这个请求可能最终通过北桥，到达南桥。当 CPU reset 时，其第一条指令请求的地址为 FFFFFFF0h，此时内存并未初始化，这个请求会从 bus 上被转发到南桥，南桥最终将其重定向到 LPC/SPI 的 ROM 上，所以 bootblock 阶段，所有的代码都是从 ROM 上读取的。这也是从概念上区分 bootblock/PEI 和 POST/DXE 阶段的方法。那么，如果内存初始化过后，CPU 读取 FFFFFFF0h 的 request 是否有可能被内存接收到呢？No，在内存控制器初始化的时候，会设置一些参数，保留几段地址空间的区域给 flash rom， PCI 设备分配使用，它确保不会将内存映射到这些地址空间上。所以内存初始化过后，地址请求 FFFFFFF0h 还是会被北桥转发出去，不会 decode 到内存。但是，如果一个 PCI 设备，在南桥之前收到了这个请求，并且解码了这个地址，回复了 CPU 的请求，那么将不会从南桥的 LPC/SPI boot。

CPU reset 之 CPU mode

当 CPU 抓到第一条指令，并执行的时候，此时 CPU 处于那种模式呢？如果说处于 real mode，那么 FFFFFFF0h 的地址，早已超出了 real mode CPU 的寻址范围。也许很多人都讨论过这个问题。首先，可以确定的是：

1. 此时 CPU 并不处于 protect mode。CPU 进入 protect mode 的标志是 CR0 的 PE bit 置 1，而当 CPU 执行第一条指令时，此 Flag bit 并未置 1。

如果说此时 CPU 处于 real mode，那么此时 CS = F000h, IP=FFF0h, 按照 real mode 的寻址方式，此时形成的第一条指令的地址应该是 CS:IP = F000h * 10h + FFF0h = FFFF0h，而不是 FFFFFFF0h

那么我们来回顾一下，自从 386 以来，这里到底发生了什么？CS 不再是单个的段寄存器，而是包括 Segment selector， segment base， 和 segment limit 3 个 register 的一组寄存器。那最终 segment 的基地址是由谁决定呢？显然，Segment Base 决定着基地址的值。当 CPU 处于段寻址模式时，如果 Segment Selector 装入 F000h, 那么 CPU 会自动将 F000h*10h = F0000h 装入 Segment Base，那么最终形成的地址就是 F0000h +IP，即 Segment Base + IP 形成地址，而不是 CS*16 +IP 形成地址。然而，CPU reset 过后，虽然此时 CPU 的确处于段寻址模式，Segment Selector 的值为 F000h，但是 Segment Base register 的值却并没有按照段寻址模式的方式装入，此时 Segment Base = FFFF0000h，按照 Segment Base+IP 的方式，最终形成的地址为 FFFF0000h + FFF0h = FFFFFFF0h。这就是为何第一条指令会从 FFFFFFF0h 抓，而不是 FFFF0h 抓的原因。

那么，其实我们可以说，CPU 执行第一条指令时，它处于一种不普通的 real mode 中。

CPU reset 之种类

也许这个课题和本文的主题不是很相关，这一节将要说明的是，当系统运行的时候，能够使 CPU reset 的方法。

方法一：向 IO Port CF9 写 0Eh。这是一种非常彻底的 reset 方法，系统的 HW 会掉电后重新上电

方法二：向 IO Port CF9 写 06h。这种 reset 方法也属于 code reset，不同于方法一的是，

这种 reset 方法不会使系统设备掉电，仅仅将 CPU 和系统设备的 status 干净彻底的 reset 一下。

方法三：向 IO Port CF9 写 04h。Soft reset 方法之一

方法四：向 IO Port 92h 写 01h。 Soft reset 方法之一

方法五：向 KBC port 66h 写 FEh command。同样也是 Soft reset 方法之一，不过是通过 KBC interface 拉南桥的 KBC_RST#信号，使系统 reset。

特别的 reset：还有一种 Legacy BIOS 中常见的方法：使 CPU 直接跳转到 reset vector 去执行。这种方法严格来说，并不属于 reset，但能达到某些 reset 的效果。

信号上的差异：reset 方法三，方法四和方法五都是 soft reset，其最终的效果也相同，都是通过拉 CPU 的 Init# 信号使 CPU 进入 reset 状态，使用这种 reset 方式，一些 CPU 的 register 是没有被清除的。而 reset 方法二则是通过将 PLTRST#拉低 16 个 cycle, 使系统上所有的 device 进入 reset 状态，方法一基本和方法二一样，只是会使系统设备掉电后重启，是最彻底的 reset 方式。

EFI 与 Legacy BIOS 之差异

在安排 CPU 第一条指令上，EFI 和 Legacy BIOS 是有着很大的不同。就 Legacy BIOS 来说，放在 FFFFFFF0h 的第一条指令一般是一个 Far Jump，也就是说 CPU 在执行 Legacy BIOS code 时，会直接从 FFFFFFF0h 跳回 F Segment，回到 1MB 以下这个 Legacy BIOS 的老巢。所以 Legacy BIOS 在第一条指令以后，实际上使用南桥的 E0000~ FFFFFh 这段 Legacy Region 来把它的 request 送到 Flash ROM。 所以我们在上文中提到，chipset 无法不支持 Legacy Region，是因为 Legacy BIOS 一直会依赖着它。

对于 EFI BIOS 来说，实际上第一条指令是一条 wbinvd，之后做一些设定之后，会直接进入 protect mode。所以 EFI BIOS code 是从南桥 Region 4G 通过，并不需要 Legacy Region。未来随着 EFI BIOS 的普及，南桥也可以不再支持 Legacy Region，卸掉一部分历史的包袱。