八、 NewBluePill 内存系统

1) 分页机制

- 相关文件： nbp-0.32-public\common\Paging.c

  nbp-0.32-public\common\Paging.h

- 技术背景：

分页机制是整个 nbp 的核心之一，称其核心是因为 nbp 所使用的很多其它部分都需要分页机制的支持，比如调试部分中调试信息的存放，还有整个 nbp 的代码段，以及 nbp 安装到每个 CPU 上后为了运行 nbp 所分配的堆栈区，这些内存（页）全部需要映射到 nbp 的分页机制上。

nbp 的分页机制是利用了 64 位系统特有的四级页表分页机制。AMD 和 Intel 对应的硬件实现基本相似。不过现在仍需定位 nbp 在哪里赋值给 EFER 和 CR4 寄存器，以及它赋的什么值，这影响到 nbp 如何启用相应的地址翻译机制。对于 AMD 的 CPU 来说，其各级页表寻址翻译总体过程如图 8.1 所示，每级 Virtual address 的地址结构要求可参考 AMD64 Architecture Programmer's Manual, Volume 2: System Programming 的第 131 页的四张图。Long Mode 的地址翻译机制还需要注意 CR3 的寄存器内容，在长模式下(Long Mode)，CR3 的寄存器内容包含如下部分（见图 8.2）：

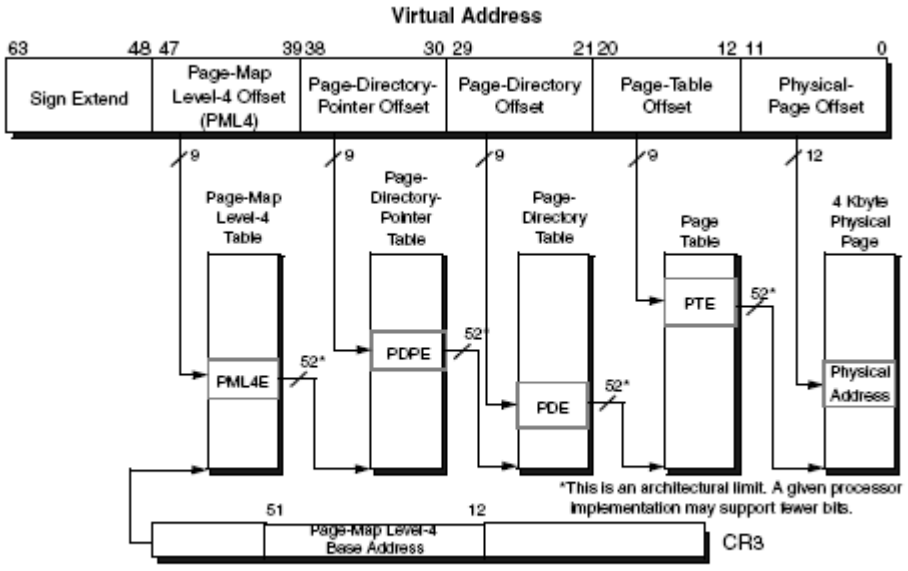

Figure 5-17. 4-Kbyte Page Translation—Long Mode

图 8.1 AMD 长模式(Long Mode)地址——4K 页翻译图

1. 页表基址域(Table Base Address Field)，Bit 51-12，这 40 位指向 PML4 页表基地址，PML4 页表必须是页对齐的。
2. 高页写穿位（Page-Level Writethrough (PWT) Bit），Bit 3，表明最高级别的页表

使用写回(Writeback)还是写穿(Writethough)策略

3. 高页可缓存位(Page-Level Cache Disable (PCD) Bit)，Bit 4，表明最高级别的页表是否可缓存。
4. 保留位(Reserved Bits)，保留位在写 CR3 寄存器时候应该清零。

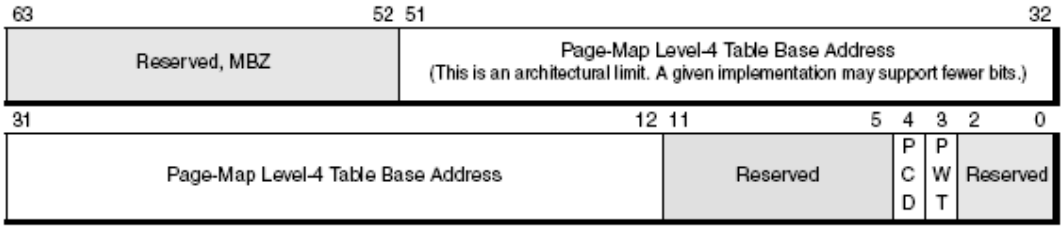

Figure 5-16. Control Register 3 (CR3)—Long Mode

图 8.2 AMD 长模式(Long Mode)中 CR3 寄存器内容

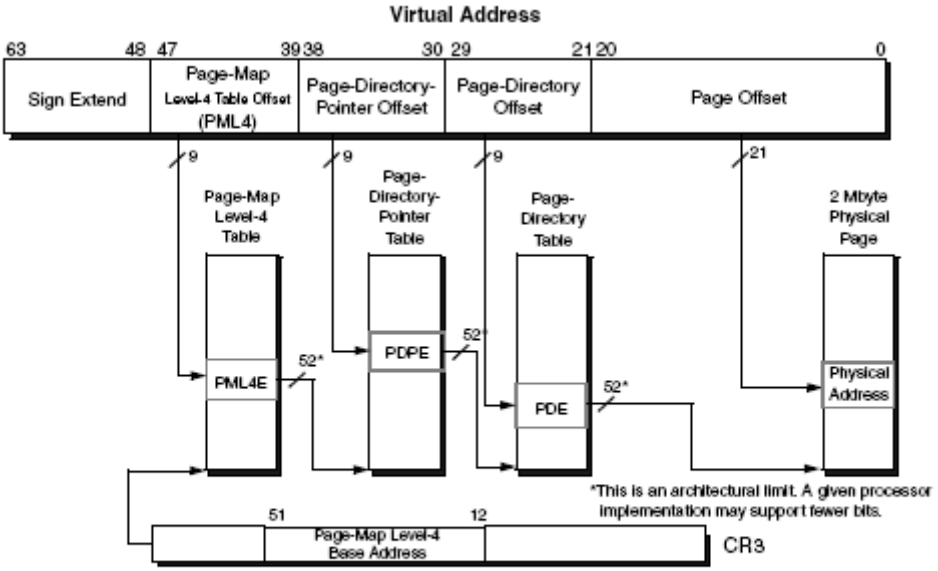

Figure 5-22. 2-Mbyte Page Translation—Long Mode

图 8.3 AMD 长模式(Long Mode)地址——2M 页翻译图

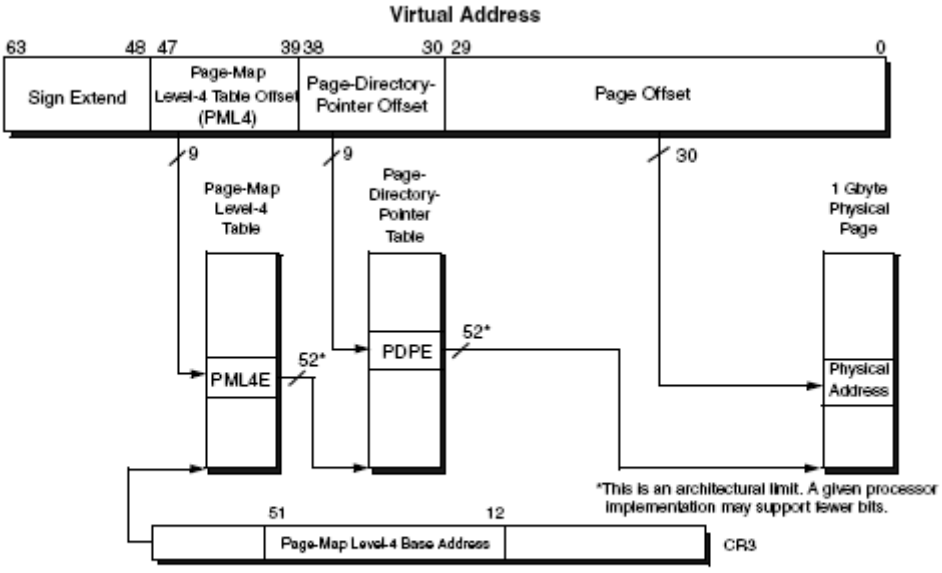

Figure 5-26. 1-Gbyte Page Translation—Long Mode

图 8.4 AMD 长模式(Long Mode)地址——1G 页翻译图

同时 AMD 也支持 2M 和 1G 页，通过分别挂载到 PD 和 PDP 即可实现，分别为 21 位寻址和 30 位寻址，比较简单因此不再叙述。（见图 8.3 和 8.4）

而对于 Intel 的 CPU 来说，这个翻译过程被称作 IA-32e Mode Linear Address Translation （IA-32e 模式地址翻译），各级页表的名称和翻译原理均十分相像，故在此仅列出 4K 页的映射过程(见图 8.5)，具体原理可参考 Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A 中第 3-42Vol3（第 126 页 Protect-Mode Memory Management）内容

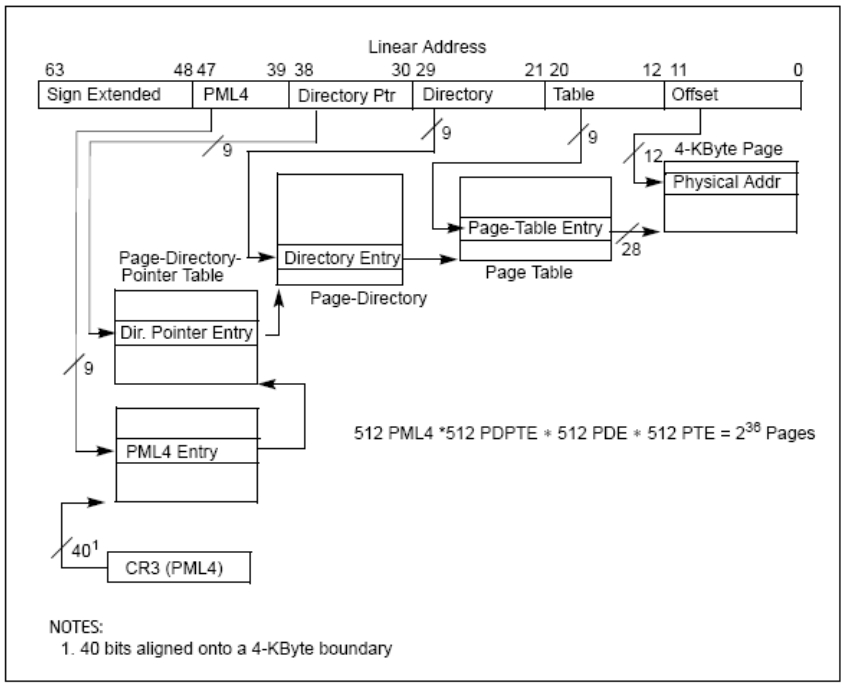

Figure 3-24. IA-32e Mode Paging Structures (4-KByte Pages)

图 8.5 Intel IA-32e 模式地址——4K 页翻译图

- 实现过程：

在谈了那么多背景知识后，我们看一看 nbp 是去使用页表地址翻译的。nbp 利用了硬件提供的页表做地址翻译，所以 nbp 在内存管理上主要做两件事情：

1. 根据硬件要求构建页表结构，并将需要挂载的页挂载到这个页表结构上。
2. 在需要的时候改变 CR3 寄存器内容，从而使 CPU 使用 nbp 的页表做地址翻译。（这个时间点仍需确定）（CR3 更新内容后怎么找到 nbp 各段地址以继续运行？）

◆ MmInitManager()方法

从 Newbp.c 的 DriverEntry 方法进入，可以看到在通过 ComInit()方法设置当前运行的 bluepill 唯一标示 ID 后，立即调用了 MmInitManager()方法来构建页表，这个函数也是 nbp 内存管理部分的入口函数，其作用是确定 PML4 页表的基地址（第 621 行）并初始化从 PML4 到 PT 每级一个页表，当然此时也初始化好了他们之间的映射关系，后者过程是通过 MmCreateMapping()方法和 MmUpdatePageTable()方法递归完成的。

可以观察到在 MmInitManager 中出现了两组地址翻译的操作，一个是从 OS 的虚拟地址到内存物理地址的映射，另一个是 VM 的虚拟地址（或者说是 nbp 的虚拟地址）到内存物理地址的映射。在 nbp 中，HostAddress 指 nbp 中 VM 寻址用的虚拟地址，PhysicalAddress 指物理内存地址，而 GuestAddress 是指 OS 寻址用的虚拟地址。在后面我们可以看到，nbp 为了实现方便，把页的 GuestAddress 作为 HostAddress 直接挂在了 nbp 的页表上。

此外，nbp 为了方便管理这套页表（主要是为了方便查找和释放某页），利用 Windows 提供的双向链表将 nbp 自己所有的页表信息和页信息保存了起来并且顺序挂在这个双向链表上，实现这个过程的方法是 MmSavePage(),

这里有一个细节就是每级页表所对应的 HostAddress，依次定义如下：（定义在 \nbp-0.32-public\commoncommon.h 的第 122 行）

```
#define   PML4_BASE    0xFFFFF6FB7DBED000
#define   PDP_BASE     0xFFFFF6FB7DA00000
#define   PD_BASE      0xFFFFF6FB40000000
#define   PT_BASE      0xFFFFF68000000000
```

此处的几个地址是固定的地址，这几个地址的计算可参考参考书目一章中的[4][5][6]，此处仅列出 Windows x64 address space layout（图 8.6，512GB 4 level Page table map 行使我们所关心的，这四个地址全在那段中）

当然更进一步的搜索发现在 xen 中也有完全相同的上面四个数字，因此如果我们是开发自己的系统，可以直接拿这几个 base address 来用，常量不变。

◆ MmSavePage ()方法

作用是将 nbp 分配出来的某个页表/页的信息保存起来并顺序挂在 g_PageTableList 这个双向链表上，这个信息被存在 ALLOCATED_PAGE 结构体中，主要记录了一个页表/页的物理地址，在 OS 中的虚拟地址（Guest Address,某些时候使用以及释放该页时要借助于 OS 的帮忙），在 nbp 内部的虚拟地址（Host Address,对于页来说这个地址就是其在 OS 中的虚拟地址），分配类型(Allocation Type),页数量（uNumberOfPages，nbp 只支持整页分配，因此这项必为整数），标志位（Flags，标记了关于该页的其它信息）

这里关于 PhysicalAddress 只取中间的 40 位，而 HostAddress 要取高 52 位的一个原因是对于 PhysicalAddress 现在的内存远远不需要用 64 位来表示，此处最细粒度是 4K 页，因此低 12 位在此处是不用存的，因为 CR3 中要求接受物理内存地址的中间 40 位，

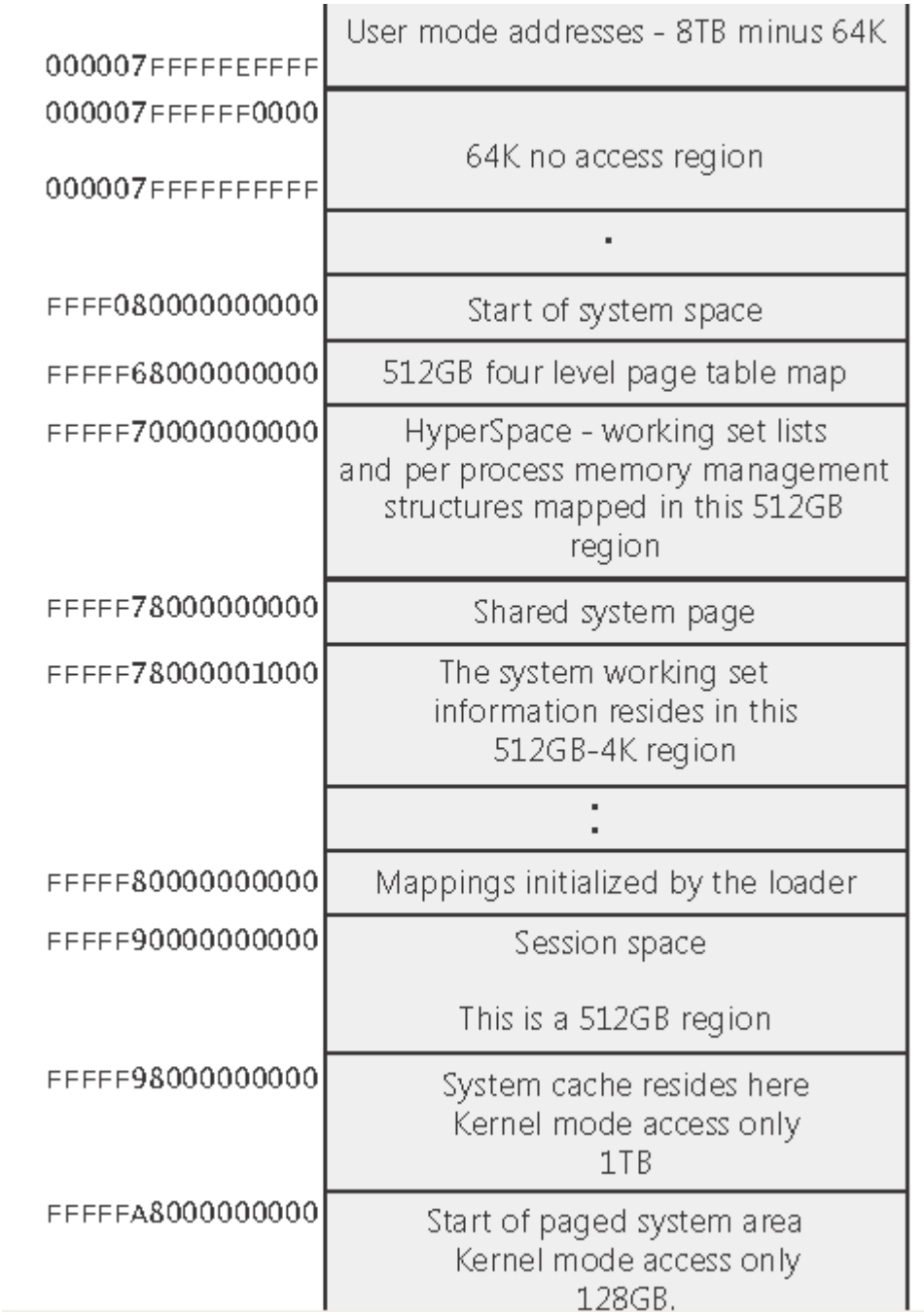

图 8.6 Windows x64 结构地址空间图(x64 address space layout)

因此只需满足 CR3 的要求即可，当然如果为了以后 nbp 也能跑起来，CR3 高端部分开几位这里也就跟着开几位就可以了。

HostAddress 高 12 位必须保留，因为规范中规定这部分是 Sign extend 第 47 位得来的

关于分配类型(AllocationType)，在 nbp 中一共有三种内存分配类型：

- PAT_POOL 同 Windows 中的 a block of pool memory.
- PAT_CONTIGUOUS 同 Windows 中的 Contiguous Memory
- PAT_DONT_FREE 分配出来的多页内存，这种表示除了第一页外其它页的类型：此空间已被使用。（没什么别的含义）

关于标志（Flags）,在 nbp 中对页表的标志有 5 个：

- AP_PAGETABLE：表明该信息指一个页表的信息
- AP_PT：该页表是 PT 级页表
- AP_PD：该页表是 PD 级页表
- AP_PDP：该页表是 PDP 级页表
- AP_PML4：该页表是 PML4 级页表

- MmCreateMapping ()方法
  这个方法作用是创建 PhysicalAddress 和 VirtualAddress（也就是 HostAddress）之间的映射，实际上构造完的结果可以使得用 VirtualAddress 去寻址 PhysicalAddress
  (未完待续)
- MmSavePage ()方法
- MmSavePage ()方法