该章节内容如未标注则来自于 **深入理解linux内核(中文第三版)**

程序员在访问内存的时候需要用到 `内存地址` ，在32位机器上它是一个 `uint32_t` 类型的数据，在64位机器上则是 `uint64_t` 的数据。在80x86处理器上有三种不同的地址：

- 逻辑地址(logical address)

逻辑地址在分段管理内存时非常常见，它促使MS-DOS或Windows程序员把内存分为若干段进行管理。每一个逻辑地址都由一个段和偏移量组成，偏移量指明了从段开始的地方到实际地址之间的距离。

- 线性地址(linear address)(也称虚拟地址 virtual address)

现代操作系统和处理器使用虚拟内存技术来实现内存隔离和扩展，有了这样的抽像以后，一个程序就可以使用比真实物理地址大得多的地址空间，甚至多个进程可以使用相同的地址。

- 物理地址(physical address)

用于内存芯片级内存单元寻址，在现代计算机系统中，操作系统和处理器使用虚拟内存技术来管理内存。程序使用虚拟地址访问内存，而这些虚拟地址通过内存管理单元（MMU）转换为物理地址。虚拟地址和物理地址之间的映射由页表等数据结构完成。

- DRAM地址
  用于在DRAM存储芯片上索引物理存储单元。

内存控制单元(MMU)通过一种称为分段单元(segmentation unit)的硬件电路把一个逻辑地址转换成线性地址;接着,第二个称为分页单元(paging unit)的硬件电路把线性地址转换成一个物理地址。

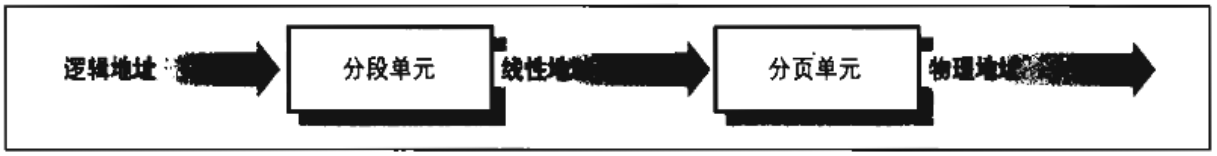


图 2-1：逻辑地址转换

逻辑地址到线性地址

第一部分是将逻辑地址通过分段单元转化成线性地址，在**linux系统中对分段的使用非常有限**，原因有以下几点：

1. 当所有的程序都使用相同的段寄存器值时内存管理更加简单。
2. Linux的设计目标就是能在各种处理器上运行，有些处理器(比如RISC)对分段的支持非常有限。

因此linux只在80x86处理器上才需要使用分段(2.6版本)。运行在**用户态的所有Linux进程都使用一对相同的段来对指令和数据寻址**，这两个段就是所谓的用户代码段和用户数据段。类似地,运行在**内核态的所有Linux进程都使用一对相同的段对指令和数据寻址**,它们分别叫做内核代码段和内核数据段。

表 2-3：四个主要的 Linux 段的段描述符字段的值

段	Base	G	Limit	S	Type	DPL	D/B	P
用户代码段	0x00000000	1	0xfffff	1	10	3	1	1
用户数据段	0x00000000	1	0xfffff	1	2	3	1	1
内核代码段	0x00000000	1	0xfffff	1	10	0	1	1
内核数据段	0x00000000	1	0xfffff	1	2	0	1	1

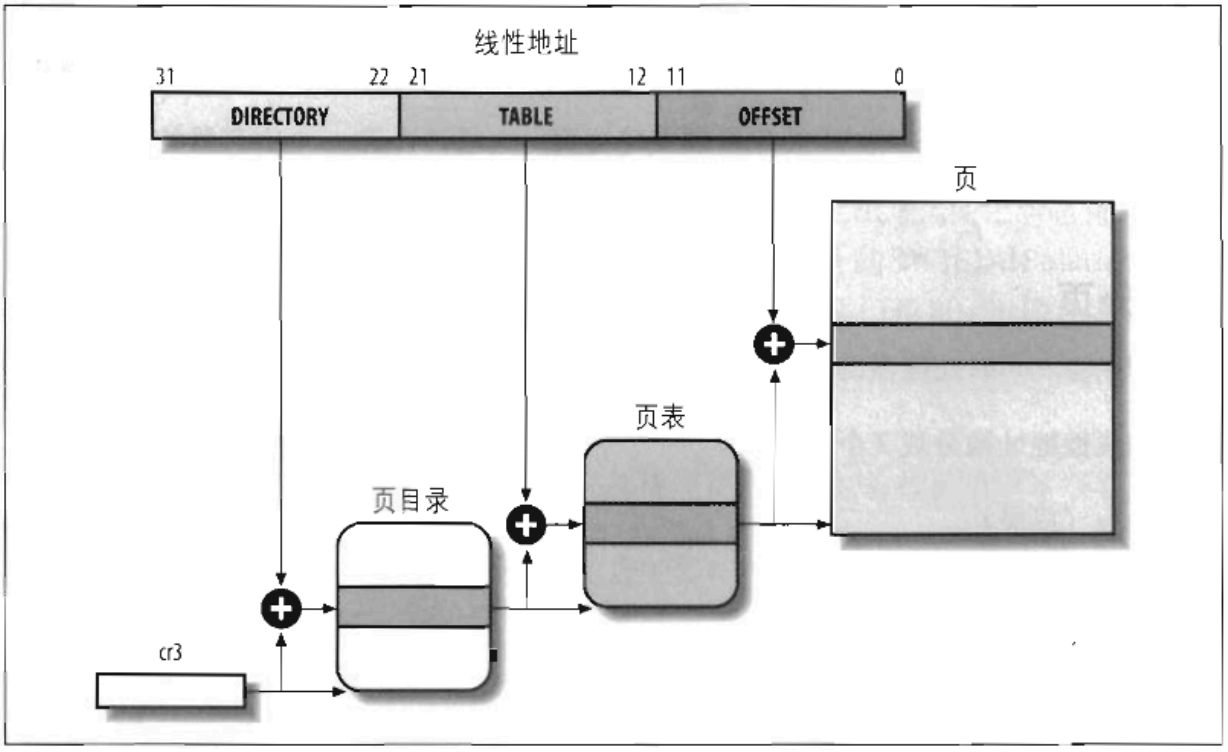
与段相关的线性地址从0开始,这就意味着在用户态或内核态下的所有进程可以使用相同的逻辑地址。由于所有段都从0x00000000开始,所以在**Linux下逻辑地址与线性地址是一致的**,即逻辑地址的偏移量字段的值与相应的线性地址的值总是是一致的。

线性地址到物理地址

把线性地址转换成物理地址的工作由分页单元(paging unit)完成。为了效率起见,线性地址被分成以固定长度为单位的组,称为页(page)。页内部连续的线性地址被映射到连续的物理地址中。这样,内核可以指定一个页的物理地址和其存取权限,而不用指定页所包含的全部线性地址的存取权限。

分页单元把所有的RAM分成固定长度的页框(page frame)(有时称为物理页)。每一个页框包含一个页(page),页框是主存的一部分,因此也是一个存储区域。 把线性地址映射到物理地址的数据结构称为页表(page table)。页表存放在主存中,并在启用分页单元之前必须由内核对页表进行适当的初始化。从80386开始,所有的80x86处理器都支持分页,它通过设置cr0寄存器的PG标志启用。当PG=0时,线性地址就被解释成物理地址。

以32位系统为例，常见的4KB大小的页中一个32位地址一般由三个区域组成，从高到低依次是 目录(Directory,10位)、页表(Table, 10位)、页偏移(Offset, 12位)，正在使用的页目录的物理地址存放在控制寄存器cr3中，具体转化方式如图所示：



在64位Linux中如果只使用上面这种二级页表方案就会造成极大的资源浪费(新建一个进程就需要分配出页目录所需的空间)，因此它使用了多级页表方案

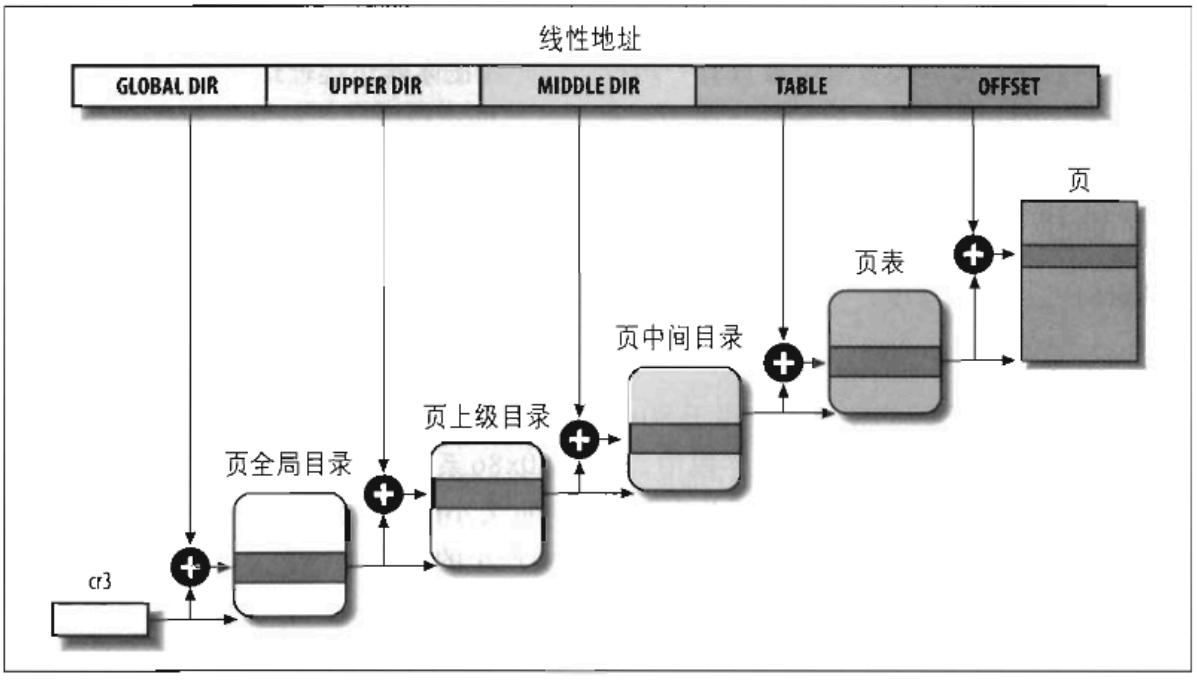


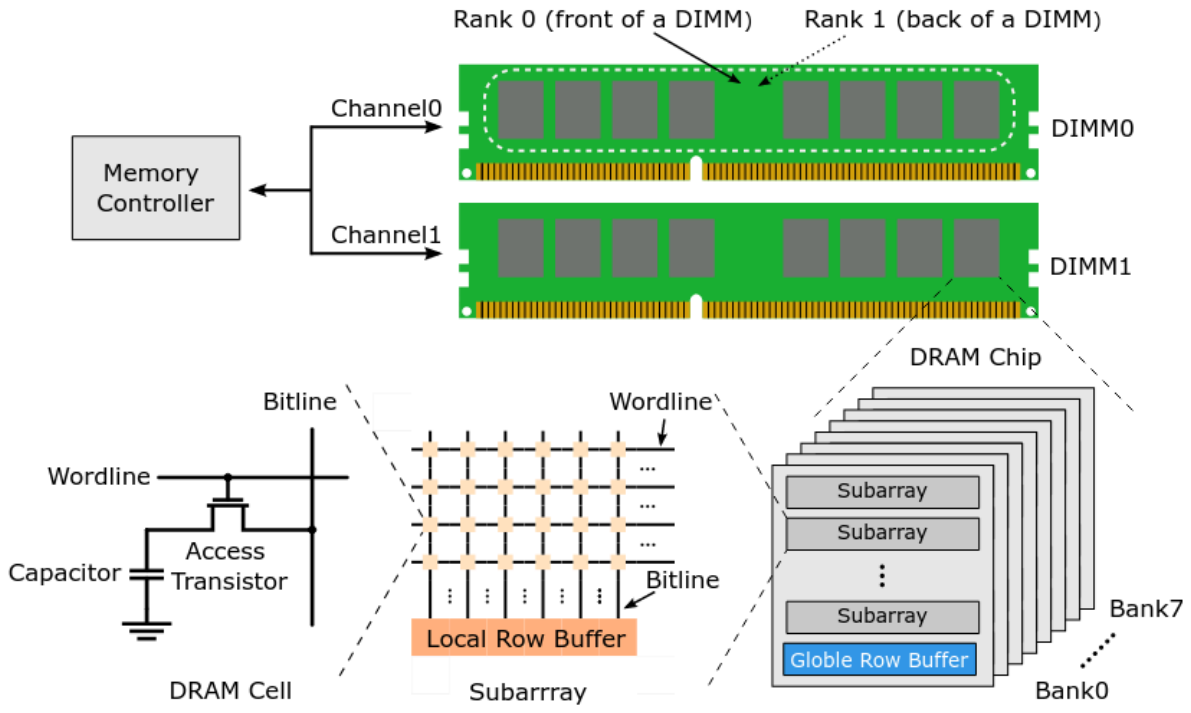
图 2-12：Linux 分页模式

物理地址转换到DRAM地址

该部分内容来自于 ZENHAMMER: Rowhammer Attacks on AMD Zen-based Platforms、SoK: Rowhammer on Commodity Operating Systems、DRAMDig: A Knowledge-assisted Tool to Uncover DRAM Address Mapping

DRAM芯片结构

目前常见的DRAM模块是双列直插式内存模块(DIMM)，`存储器控制器( MC )`通过通道与DIMM通信，每个通道由命令总线、地址总线和数据总线组成。一个DIMM有一个或多个rank(**一般看内存芯片有几面**)。一个rank有一组内存芯片，每个芯片有若干个bank。在数据总线为64位宽的典型情况下，一个rank有8个芯片，一个芯片上有8个bank来响应数据总线的请求。一个bank由多个子阵列和一个全局行缓冲区组成。每个子阵列是一个具有局部行缓冲区的二维单元格数组，存储最近访问的特定行单元格的数据。一行中的单元格通过`Wordline`水平连接。列中的单元格通过`Bitline`垂直连接到本地行缓冲区。本地行缓冲器通过全局位线与全局行缓冲器相连。一个单元由一个作为开关的存取晶体管和一个存储单个比特的电容器组成。



DRAM芯片寻址

DRAM芯片寻址包括两个主要步骤，具体细节主要由CPU的内存控制器决定，因此在不同架构的CPU上会有所区别

1. MC将物理地址映射到逻辑DRAM地址。逻辑地址是一个3元组(bank索引、row索引、column索引)，其中bank索引包括channel、DIMM和rank。
2. 将逻辑DRAM地址重新映射到物理DRAM地址。

在MC视图中彼此相邻的逻辑行在DRAM中很可能在物理上不相邻，但这两步对于RowHammer而言十分关键且没有公开的文档，因此大部分研究都是采用逆向工程来获取映射方法。

大部分的芯片进行映射要么使用物理地址的单个位，要么使用多个位的 xor 组合来计算实际的索引。通过逆向工程可以获取到具体使用的xor函数，如图所示(因CPU和内存而异)

Table 3. Reverse engineered address mappings and offsets for different DRAM configurations. All memory configurations are single-channel, single-DIMM, with the tuple indicating the DIMM's geometry (#ranks, #bank groups, #banks per bank group, #rows).

Sys.	Geometry (RK, BG, BA, R)	Size [GiB]	Offt. [MiB]	DRAM Address Functions			Row Bits
				Rank (RK)	Bank Group (BG)	Bank Address (BA)	
Z_+	$(1, 4, 4, 2^{16})$	8	1024	n/a	0x088883fc0, 0x111104000	0x022228000, 0x044450000	32 – 17
	$(2, 4, 4, 2^{16})$	16	1024	0x3fffe0000	0x111103fc0, 0x222204000	0x044448000, 0x088890000	33 – 18
	$(2, 4, 4, 2^{17})$	32	1024	0x7fffe0000	0x111103fc0, 0x222204000	0x444448000, 0x088890000	34 – 18
Z_2	$(1, 4, 4, 2^{16})$	8	512	n/a	0x088883fc0, 0x111104000	0x022228000, 0x044450000	32 – 17
	$(2, 4, 4, 2^{16})$	16	512	0x3fffe0000	0x111103fc0, 0x222204000	0x044448000, 0x088890000	33 – 18
	$(2, 4, 4, 2^{17})$	32	512	0x7fffe0000	0x111103fc0, 0x222204000	0x444448000, 0x088890000	34 – 18
Z_3	$(1, 4, 4, 2^{16})$	8	768	n/a	0x022220100, 0x044440200	0x088880400, 0x111100800	32 – 17
	$(2, 4, 4, 2^{16})$	16	768	0x3fffe0000	0x044440100, 0x088880200	0x111100400, 0x222200800	33 – 18
	$(2, 4, 4, 2^{17})$	32	768	0x7fffe0000	0x444440100, 0x088880200	0x111100400, 0x222200800	34 – 18

对于AMD的ZEN架构芯片而言需要进行些许改进，因为它的地址映射并非线性，还需要减去一个特定的常量偏移来消除这种非线性，因为有一部分的内存交与PCI设备使用，物理地址空间会重新映射。

一般而言， 一个row 有 2^{16} 个字节，也就是 `256k`，但是不同cpu、内存可能会有些许不同。

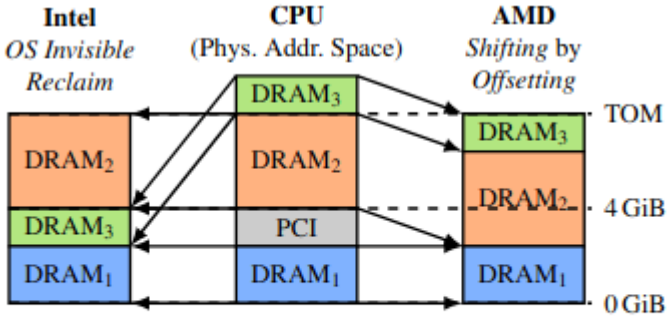


Figure 3. Remapping of higher address ranges to unused parts of physical memory on Intel and AMD CPUs. The Top of Memory (TOM) is the system's highest addressable memory location.

System	PCI Range [MiB]	Offset [MiB]
Z_+	3072 – 4048	1024
Z_2	3584 – 3968	512
Z_3	3328 – 4076	768

Table 2. Primary PCI memory mappings and detected physical address offsets, i.e., difference between 4 GiB and the PCI mapping's start address.