```
Booting Linux on physical CPU 0xffff00
Linux version 4.2.8-yocto-standard (walterzh@walterzh-Precision-T1650) (gcc version 4.9.3 (GCC) ) #1 SMP PREEMPT Tue Jun 21 15:32:28 CST 2016
CPU: AArch64 Processor [410fd034] revision 4
Detected VIPT I-cache on CPU0
alternatives: enabling workaround for ARM erratum 845719
efi: Getting EFI parameters from FDT:
efi: UEFI not found.
PERCPU: Embedded 15 pages/cpu @ffffffc03ff50000 s22400 r8192 d30848 u61440
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 257024
Kernel command line: console=ttyS0,115200n8 earlyprintk=serial,ttyS0,115200 root=/dev/mmcblk1p2 uio_pdrv_genirq.of_id=generic-uio rootwait
PID hash table entries: 4096 (order: 3, 32768 bytes)
Dentry cache hash table entries: 131072 (order: 8, 1048576 bytes)
Inode-cache hash table entries: 65536 (order: 7, 524288 bytes)
software IO TLB [mem 0x3ac00000-0x3ec00000] (64MB) mapped at [ffffffc03ac00000-ffffffc03ebfffff]
Memory: 952044K/1044480K available (4823K kernel code, 282K rwdata, 1788K rodata, 244K init, 560K bss, 92436K reserved, 0K cma-reserved)
Virtual kernel memory layout:
    vmalloc : 0xffffff8000000000 - 0xffffffbdbfff0000   (   246 GB)
    vmemmap : 0xffffffbdc0000000 - 0xffffffbfc0000000   (     8 GB maximum)
              0xffffffbdc0000000 - 0xffffffbdc1000000   (    16 MB actual)
    fixed   : 0xffffffbffa7fd000 - 0xffffffbffac00000   (  4108 KB)
    PCI I/O : 0xffffffbffae00000 - 0xffffffbffbe00000   (    16 MB)
    modules : 0xffffffbffc000000 - 0xffffffc000000000   (    64 MB)
    memory  : 0xffffffc000000000 - 0xffffffc040000000   (  1024 MB)
      .init : 0xffffffc0006f8000 - 0xffffffc000735000   (   244 KB)
      .text : 0xffffffc000080000 - 0xffffffc0006f7474   (  6622 KB)
      .data : 0xffffffc000738000 - 0xffffffc00077e800   (   282 KB)
```

in Documentation/arm64/memory.txt

```
This document describes the virtual memory layout used by the AArch64
Linux kernel. The architecture allows up to 4 levels of translation
tables with a 4KB page size and up to 3 levels with a 64KB page size.
```

```
CONFIG_ARM64_4K_PAGES=y
#CONFIG_ARM64_64K_PAGES

CONFIG_PGTABLE_LEVELS=3
```

pegmatite还是使用了3级页表，page size = 4K

```
AArch64 Linux uses either 3 levels or 4 levels of translation tables
with the 4KB page configuration, allowing 39-bit (512GB) or 48-bit
(256TB) virtual addresses, respectively, for both user and kernel. With
64KB pages, only 2 levels of translation tables, allowing 42-bit (4TB)
virtual address, are used but the memory layout is the same.
```

```
CONFIG_ARM64_VA_BITS_39=y
# CONFIG_ARM64_VA_BITS_48 is not set
CONFIG_ARM64_VA_BITS=39
```

使用了39 bit virtual address, 64 - 39 = 25，最高位25位全为1.
2 ^ 39 = 512G

> memory : 0xffffffc000000000 - 0xffffffc040000000 (1024 MB)
> physical memory (1G)被mapping到了 `0xffffffc000000000` 开始的virtual address.
>
> ffffff8000000000 ffffffffffffffff 512GB kernel

用户空间与内核空间以 `0xffffffc000000000` 为界.
而[ffffff8000000000, 0xffffffc000000000)的kernel space (256G)没有使用

```
root@granite2v8:~# cat /proc/self/maps
00400000-0048f000 r-xp 00000000 b3:22 383                                /bi
n/busybox.nosuid
0049e000-004a0000 rw-p 0008e000 b3:22 383                                /bi
n/busybox.nosuid
004a0000-004a2000 rw-p 00000000 00:00 0                                  [he
ap]
7fa6ba1000-7fa6cd1000 r-xp 00000000 b3:22 19                             /li
b64/libc-2.21.so
7fa6cd1000-7fa6ce0000 ---p 00130000 b3:22 19                             /li
b64/libc-2.21.so
7fa6ce0000-7fa6ce4000 r--p 0012f000 b3:22 19                             /li
b64/libc-2.21.so
7fa6ce4000-7fa6ce6000 rw-p 00133000 b3:22 19                             /li
b64/libc-2.21.so
7fa6ce6000-7fa6cea000 rw-p 00000000 00:00 0
7fa6cea000-7fa6d7b000 r-xp 00000000 b3:22 72                             /li
b64/libm-2.21.so
7fa6d7b000-7fa6d8b000 ---p 00091000 b3:22 72                             /li
b64/libm-2.21.so
7fa6d8b000-7fa6d8c000 r--p 00091000 b3:22 72                             /li
b64/libm-2.21.so
7fa6d8c000-7fa6d8d000 rw-p 00092000 b3:22 72                             /li
b64/libm-2.21.so
7fa6d8d000-7fa6da9000 r-xp 00000000 b3:22 89                             /li
b64/ld-2.21.so
7fa6daf000-7fa6db1000 rw-p 00000000 00:00 0
7fa6db5000-7fa6db6000 rw-p 00000000 00:00 0
7fa6db6000-7fa6db7000 r--p 00000000 00:00 0                              [vv
ar]
7fa6db7000-7fa6db8000 r-xp 00000000 00:00 0                              [vd
so]
7fa6db8000-7fa6db9000 r--p 0001b000 b3:22 89                             /li
b64/ld-2.21.so
7fa6db9000-7fa6dbb000 rw-p 0001c000 b3:22 89                             /li
b64/ld-2.21.so
7fdd8a1000-7fdd8c2000 rw-p 00000000 00:00 0                              [st
ack]
```

但从user space application的memory layout看

> 0000000000000000 0000007fffffffff 512GB user

在virtual address为39-bit的情况下，最多支持多大的physical memory?
由于physical memory address zero被映射到 `0xffffffc000000000` ,所以采用这种memory mapping的kernel最多能管理256G memory.

整个64-bit的虚拟空间被分成了 3 部分。

分配状况	范围	用处
user space	最低端的512G	application space
kernel space	最顶端的256G	Linux kernel活动空间
kernel space	次顶端的256G	reserved
中间巨大的4G个T - 1T	reserved	无法访问 |

```
AArch64 Linux memory layout with 4KB pages + 3 levels:

Start           End         Size        Use
-----------------------------------------------------------------------
0000000000000000    0000007fffffffff     512GB      user
ffffff8000000000    ffffffffffffffff     512GB      kernel


AArch64 Linux memory layout with 4KB pages + 4 levels:

Start           End         Size        Use
-----------------------------------------------------------------------
0000000000000000    0000ffffffffffff     256TB      user
ffff000000000000    ffffffffffffffff     256TB      kernel


AArch64 Linux memory layout with 64KB pages + 2 levels:

Start           End         Size        Use
-----------------------------------------------------------------------
0000000000000000    000003ffffffffff       4TB      user
fffffc0000000000    ffffffffffffffff       4TB      kernel


AArch64 Linux memory layout with 64KB pages + 3 levels:

Start           End         Size        Use
-----------------------------------------------------------------------
0000000000000000    0000ffffffffffff     256TB      user
ffff000000000000    ffffffffffffffff     256TB      kernel
```

前两者是39-bit，后两者是48-bit．
目前pegmatite使用的是39-bit中的前者。即 3 级页表。