## Tutorial notes

Here is tutorial

<https://david942j.blogspot.com/2018/10/note-learning-kvm-implement-your-own.html>

////////////////////////////

All communication with KVM is done by the ioctl syscall, which is usually used for getting and setting device status

///////////////////////////

1. Put assembled code on user memory region, set up vCPU's registers such as rip
2. Run and handle exit reasons. while(1) { ioctl(vcpufd, KVM\_RUN, 0); ... }

//////////////////////////

Notice that we can create multiple vCPUs under one VM, and with multi-threads we can emulate a VM with multiple CPUs. Note: by default, the vCPU runs in [**real mode**](https://en.wikipedia.org/wiki/Real_mode), which only executes **16-bit assembled code**. **To run 32 or 64-bit, the page table must be set up**

////////////////////////////

CR: is control registers used for interrupt control, paging control, switching the addressing modes, coprocessor control.

CR0, CR1, CR2, CR3 these are only four control registers.

31st bit of CR0 represente paging enabled then CR3 contains the PML4T base address. If CR0 31st bit is disabled then paging is not used.

In case of 64 bit some extensions might be used.

NOTE: if 4k page size is used then we need 4 levels of page tables

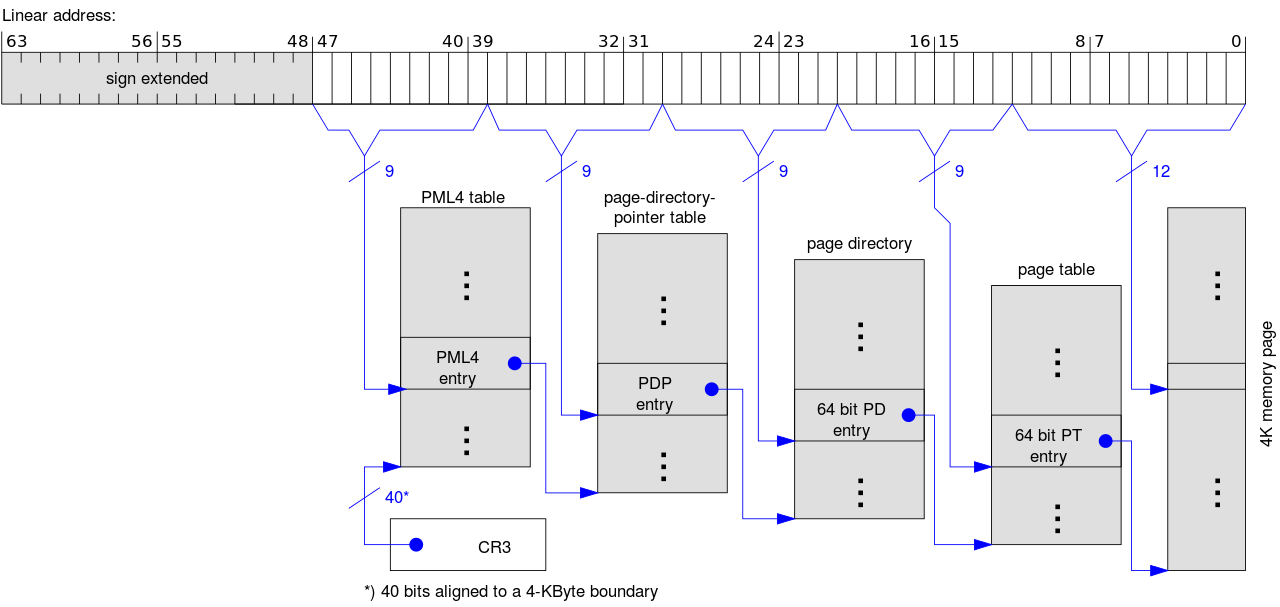
PML4T -> PDPT -> PDT -> PT -> RAM.

If 2M page size is used then PT is not used and PDT points to RAM.

PT - page table. PDT - page directory table.

PDPT- page directory pointer table.

PML4T- page map level 4 is 4rth level page table.



This is using 4KB pages hence 2^9 entries in each page. So 9 bits are used for accessing the physical address of the next level page table and in the end 12bits offset is used to get 1 byte of data. Because memory is byte addressable so 12 bits are required whereas PTE size is 3 byte hence 9 bits are required to get entry because page size is 2^12B. PTE = 2^12B / 2^9 = 3B.

/////////////////////////////

Segment selector is segment register(which has address of segment).

First few bits of logical address is the segment selector. Because we use segmented paging and use segments for code,data, and stack part of the program then we use paging in each segment.

Google it is still not crystal clear.

## General notes

* We are building hardware assisted hypervisor means processors should have VMX mode for execution.

<https://unix.stackexchange.com/questions/566243/explain-kvm-userspace-memory-region-structure>

struct kvm\_userspace\_memory\_region region = {

.slot = 0,

.guest\_phys\_addr = 0x1000,

.memory\_size = 0x1000,

.userspace\_addr = (uint64\_t)mem,

};

ioctl(vmfd, KVM\_SET\_USER\_MEMORY\_REGION, &region);

Mmap

"Copy on write" means more or less what it sounds like: everyone has a single shared copy of the same data *until it's written*, and then a copy is made. Usually, copy-on-write is used to resolve concurrency sorts of problems.

## 

## <https://github.com/soulxu/kvmsample/blob/master/main.c>

This code has used io.port to read/write.

See the code you need to write at kvm\_run vcpu .io.offset location to access.

( I think And for large data transfer use a segment in guest and allocate separate address to transfer data.)

## 

## 

## Doubt

* We stored code segment at the beginning of guest memory. And set the rip to 0, and set the rsp but what about heap memory where is it we are not supporting heap in this?

## Assembly Programming

This is for GCC compiler which is most popular in OS world.

% is used for register name

%% used for C variable name(when you want to use C variable in assembly code).