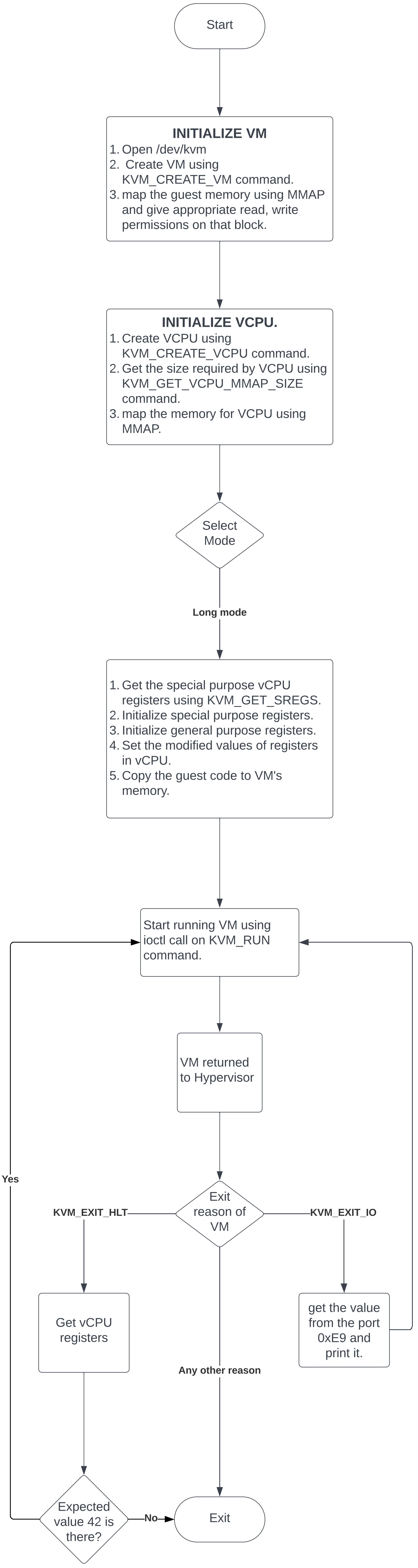
Start
INITIALIZE VM
1. Open /dev/kvm
2. Create VM using KVM_CREATE_VM command.
3. map the guest memory using MMAP and give appropriate read, write permissions on that block.
INITIALIZE VCPU.
1. Create VCPU using KVM_CREATE_VCPU command.
2. Get the size required by VCPU using KVM_GET_VCPU_MMAP_SIZE command.
3. map the memory for VCPU using MMAP.
Select Mode
Long mode
1. Get the special purpose vCPU registers using KVM_GET_SREGS.
2. Initialize special purpose registers.
3. Initialize general purpose registers.
4. Set the modified values of registers in vCPU.
5. Copy the guest code to VM's memory.
Start running VM using ioctl call on KVM_RUN command.
VM returned to Hypervisor
Exit reason of VM
KVM_EXIT_HLT
KVM_EXIT_IO
Yes
Get vCPU registers
get the value from the port 0xE9 and print it.
Any other reason
Expected value 42 is there?
No
Exit

KVM APIs used in Flow chart

1. To get the KVM’s API version, ioctl call on **KVM_GET_API_VERSION** is used.

   Ioctl call’s parameters:

   1. FD fo the kvm device.
   2. KVM_GET_API_VERSION command. It will retrieve the version of the KVM API.
   3. Additional parameters if any. Ohterwise ‘0’ is given.

2. To create the VM, ioctl call on **KVM_CREATE_VM** is used.

   Ioctl call’s Inputs:

   1. FD of the kvm device.
   2. KVM_CREATE_VM command. It creates the new VM and returns the FD of the newly created VM so that we can control that VM.
   3. Third argument is usually given as ‘0’. If we don’t want to specify additional parameters.

3. To set the memory of the VM, ioctl call on **KVM_SET_USER_MEMORY_REGION** is used.

   Ioctl call’s inputs:

   1. FD of the corresponding VM.
   2. KVM_SET_USER_MEMORY_REGION command. It will configure the memory area for the VM. specific details of that memory block will be specified in the instance of the Struct kvm_userspace_memory_region.
   3. Pointer to instance of the Struct kvm_userspace_memory_region, in which we will specify the memory size of VM, starting physical address of the VM, flags etc.

4. To create the virtual CPU inside a particular VM, ioctl call on **KVM_CREATE_VCPU** is used.

   Ioctl call’s inputs:

   1. FD of the VM in which we want to create the VCPU.
   2. KVM_CREATE_VCPU command. It will create the VCPU inside the VM specified in the first argument.
   3. Third argument is usually given as ‘0’, If we don’t want to specify additional parameters.

   Ioctl call returns the created VCPU’s FD.

5. To get the memory mapping size of the VCPU, ioctl call on **KVM_GET_VCPU_MMAP_SIZE** is used.

   Ioctl call's inputs:

   1. FD of the kvm device.
   2. KVM_GET_VCPU_MMAP_SIZE command. It asks the kernel to provide the memory mapping of a VCPU.
   3. Additional parameters if any. Otherwise given as '0' to indicate that ioctl call has no additional parameters.

   Ioctl call returns mmap size of the vcpu.

6. To run the VM, ioctl call on **KVM_RUN** is used.

   Ioctl call's inputs:

   1. FD of the vcpu of the vm, which we want to run.
   2. KVM_RUN command. It will start execution of the vcpu and it will run it until some interrupt is generated or program performs IO operation etc.
   3. Additional parameters if any. Otherwise '0' is given, it indicates no additional parameters are provided.

7. **KVM_EXIT_IO** command.

   It indicates that when VM was running, the program running on VCPU wants to perform an IO operation so VM will set the exit reason to KVM_EXIT_IO and it will return control to hypervisor, and hypervisor should handle it approprietly.

8. **KVM_EXIT_HLT** command.

   It indicates that VM has executed the HALT instruction and it'll set the exit reason as KVM_EXIT_HLT and control is given back to the hypervisor, hypervisor should handle it appropriately.

9. To get the values of the special purpose registers of the VCPU, ioctl call on **KVM_GET_SREGS** is used.

   Ioctl call's inputs:

   1. FD of the vcpu whose special purpose registers we want to retrieve.
   2. KVM_GET_SREGS command. It will ask the kernel to give the special purpose registers of the VCPU given in the first argument.
   3. Pointer to instance of the struct kvm_sregs. This structure contains variables which will hold the values of the special purpose registers. Kernel will fill the values of the registers into this structure.

10. To set the values of the registers of the vcpu, ioctl call on **KVM_SET_SREGS** is used.

    Ioctl call’s inputs:

    1. FD of the vcpu whose registers will be updated.
    2. KVM_SET_SREGS command. It will ask kernel to set the registers based on the values in the struct kvm_sregs or kvm_regs.
    3. Struct kvm_sregs, which will contain the special purpose registers values that will be written to the special purpose registers. Or struct kvm_regs, which will contain the values that needs to be written on to the general purpose registers of the vcpu.