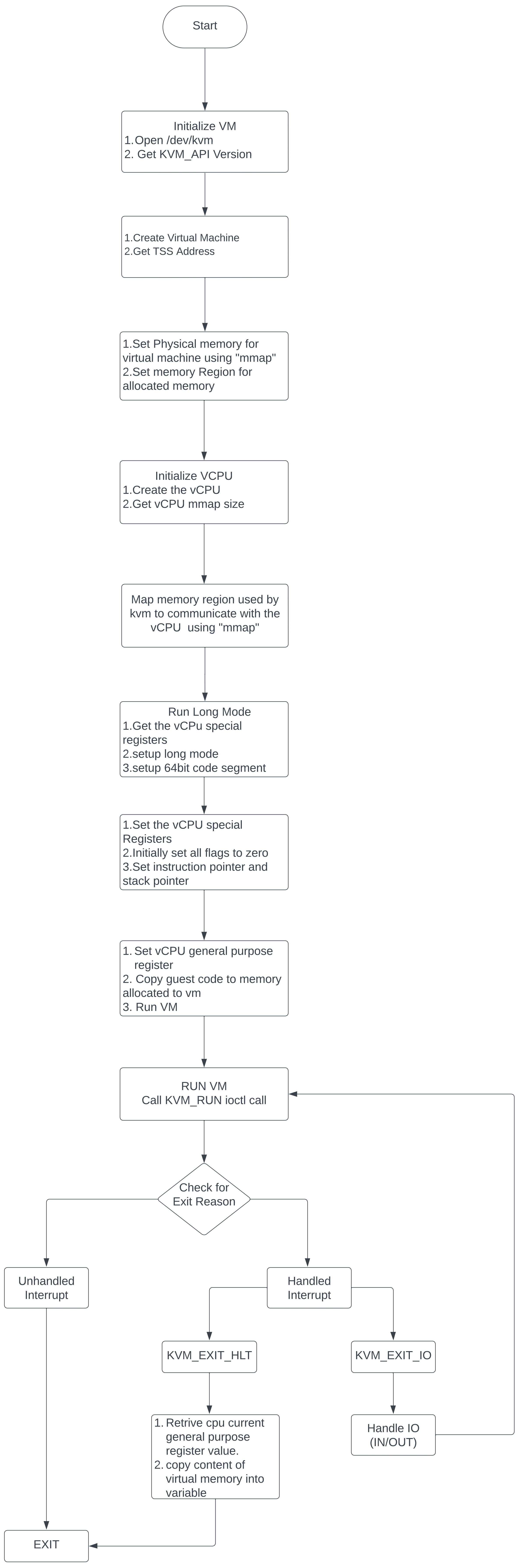
Start
Initialize VM
1.Open /dev/kvm
2. Get KVM_API Version
1.Create Virtual Machine
2.Get TSS Address
1.Set Physical memory for virtual machine using "mmap"
2.Set memory Region for allocated memory
Initialize VCPU
1.Create the vCPU
2.Get vCPU mmap size
Map memory region used by kvm to communicate with the vCPU using "mmap"
Run Long Mode
1.Get the vCPu special registers
2.setup long mode
3.setup 64bit code segment
1.Set the vCPU special Registers
2.Initially set all flags to zero
3.Set instruction pointer and stack pointer
1. Set vCPU general purpose register
2. Copy guest code to memory allocated to vm
3. Run VM
RUN VM
Call KVM_RUN ioctl call
Check for Exit Reason
Unhandled Interrupt
Handled Interrupt
KVM_EXIT_HLT
KVM_EXIT_IO
1. Retrive cpu current general purpose register value.
2. copy content of virtual memory into variable
Handle IO (IN/OUT)
EXIT

1. Open the file descriptor /dev/kvm. This is used by the user space program so that it can perform various KVM related operations such as creating ,configuring the virtual machine.The KVM_GET_API_VERSION is used to know the kvm api version .This is performed by the ioctl call which takes three parameter :

- virtual machine device fd
- KVM_GET_API_VERSION:It is macro defined in kvm header file.It specifies the ioctl operation
- 0 : no additional parameters are passed

2. To Create a virtual machine we use the ioctl call

Ioctl call is ; ioctl(vm->dev_fd,KVM_CREATE_VM,0);

- vm->dev_fd : It is a file descriptor by opening /dev/kvm
- KVM_CREATE_VM: It denotes the ioctl operation which is requested
- 0: It denotes that there is no need for additional parameter for this ioctl call

3. To map files or devices to memory we use the mmap function

It is defined as:

mmap(NULL ,mem_size, PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS | MAP_NORESERVE ,-1,0)

- NULL :ldkfn
- Mem_size: This parameter states the amount of memory to be allocated in bytes.
- PROT_READ | PROT_WRITE:This parameter specifies that the memory allocated can be read and write both.
- MAP_PRIVATE:This flag specifies that the memory will be private to the mapping process
  MAP_ANONYMOUS:This flag specifies that memory should not be associated with any kind of file instead it is set to zero.
  MAP_NORESERVE : This parameter specifies that the memory is not reserved .It is allocated on demand of the process.
- -1 : This parameter specifies the file descriptor to be mapped .In our program we have used map_anonymous so it is set to -1.
- 0: This parameter specifies the offset within the file from which the mapping should start.

4. To setup memory region for the VM we use the KVM_SET_USER_MEMORY_REGION ioctl call

Ioctl call : ioctl(vm->vm_fd , KVM_SET_USER_MEMORY_REGION , &memreg);

- vm->vm_fd: This parameter specifies the file descriptor of the virtual machine for which the memory region to be setup.
- KVM_SET_USER_MEMORY_REGION: This parameter specifies the memory region is being seup for the user memory.
- &memreg: This parameter is a structure which denotes the where to set the memory region .The structure contain the starting physical address, flags size etc.

5.To retrieve the memory mapped needed for virtual cpu which is required for communication between user and kernel space
Ioctl call is : ioctl(vm->dev_fd,KVM_GET_VCPU_MMAP_SIZE,0);

- First parameter is file descriptor of /dev/kvm
- Specific request for type of operations
- No additional parameters are required
- It returns the vcpu_mmap_size which will contain size of memory mapped area needed for vcpu state.

6.
To get the special registers value in the long mode we use the below ioctl call:
ioctl(vcpu->vcpu_fd,KVM_GET_SREGS,&sregs)

- The first parameter is the current virtual cpu fd
- The second parameter is a constant which denotes the type of operation to be performed.
- The Third parameter is a structure of type struct kvm_sregs which contain the different registers like cr0,cr1,cr2,cr3 etc.

7. To get the general purpose register we use the below ioctl call:
ioctl(vcpu->vcpu_fd,KVM_SET_REGS,&regs)

- The first parameter is the current virtual cpu fd
- The second parameter is a constant which denotes the type of operation to be performed in this case it is for GPR.
- The Third parameter is a structure of type struct kvm_sregs which contain the different registers like rax,rbx,rsi,rdi,r12,rip etc.

8. To copy the guest64 code into the vm memory area we can use the function memcpy.
memcpy(vm->mem, guest64, guest64_end_guest64)

- The first parameter is the memory area allocated for virtual machine it is a pointer.
- The second parameter is a pointer to the guest code that we want to copy into virtual machine.
- The third parameter is calculating the size of the memory region that we need to copy

9.To Instruct the hypervisor to start the virtual cpu and it also used by hypervisor to the guest code on the vcpu.
The ioctl call is: ioctl(vcpu->vcpu_fd,KVM_RUN,0);

- The first parameter is virtual cpu fd
- The second parameter is KVM_RUN which is a command that hypervisor will execute.
- The third parameter is 0 it means no additional parameter is passed.

10.To retrieve the general purpose register when the KVM_HLT instruction is executed we uses the ioctl call
ioctl(vcpu->vcpu_fd, KVM_GET_REGS,&regs);

- The First parameter is virtual cpu fd
- The second Parameter is command to retrieve the register value.
- The third parameter is the structure in which the register value will be stored.