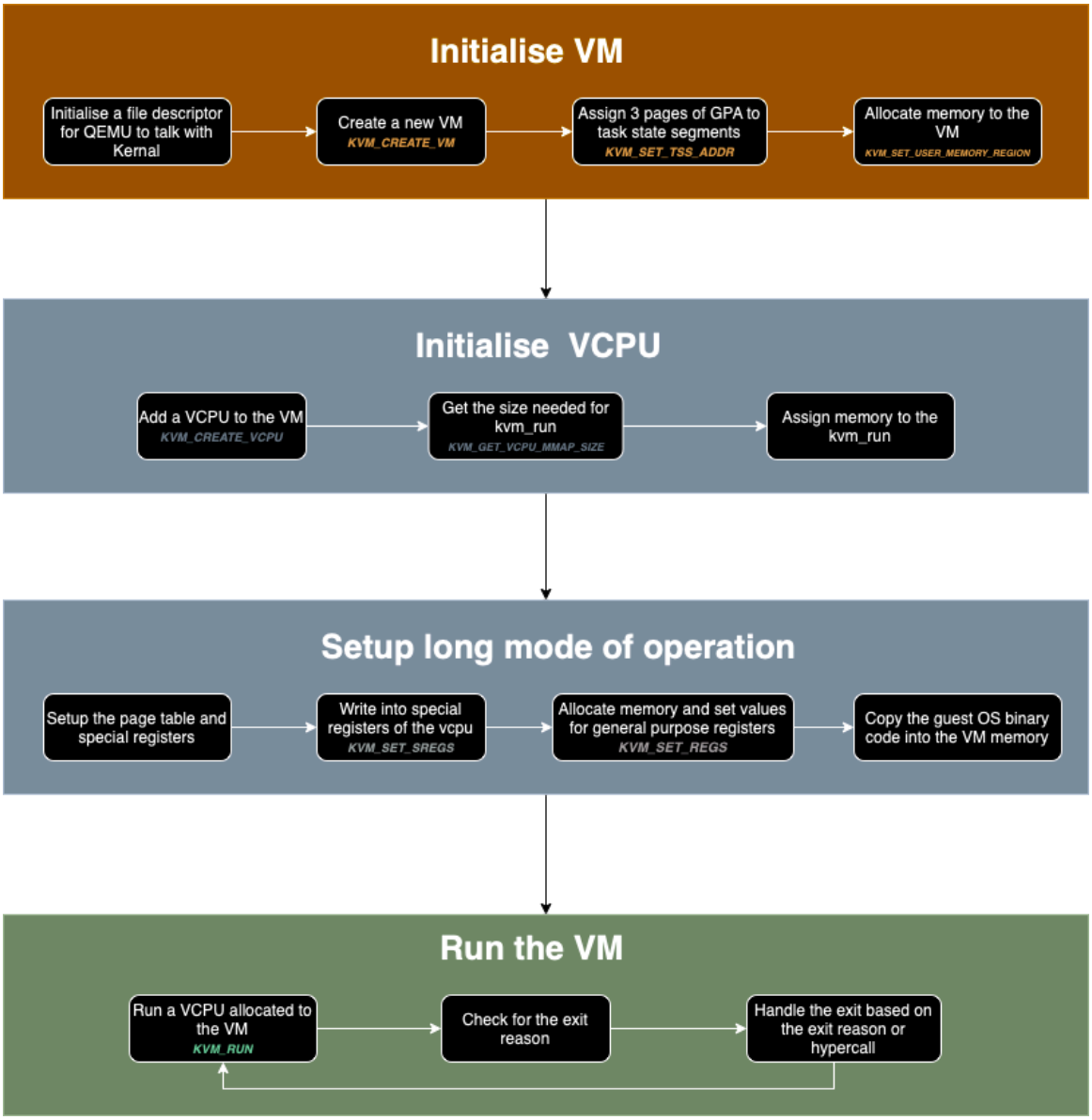

Figure 1: Flow of the execution of VM in the long mode of operation

Description of the KVM APIs mentioned in Figure 1.

1. Create a new virtual machine using KVM_CREATE_VM api.
    - KVM_CREATE_VM: It is used to create a virtual machine having no cpu and memory.
    - Parameter: dev file descriptor
    - Output: a vm fd which can be used to manage the virtual machine.

2. Assign memory for the task state segments using KVM_SET_TSS_ADDR api call.
    - KVM_SET_TSS_ADDR: This assigned 3 pages of memory in the guest physical memory space. This region is called the task state segment which is used by the guest OS to hold information regarding a task.
    - Parameter: The address where the segment should be allocated
    - Output: 0 on success, -1 on an error.
3. Allocate memory to the virtual machine using KVM_SET_USER_MEMORY_REGION api call.
    - KVM_SET_USER_MEMORY_REGION: It is used to assign guest physical memory to the VM.
    - Parameter: address of struct kvm_userspace_memory_region variable
    - Output: 0 on success, -1 on an error.
4. Create a virtual cpu for the virtual machine using KVM_CREATE_VCPU api call.
    - KVM_CREATE_VCPU: This adds a virtual cpu to the virtual machine. It takes a VCPU id as a parameter and returns a VCPU file descriptor.
    - Parameter: id of the virtual cpu (should not be greater than max_vcpu_id).
    - Output: virtual cpu file descriptor on success, -1 on an error.
5. Get the size needed for kvm_run data structure using KVM_GET_VCPU_MMAP_SIZE api call.
    - KVM_GET_VCPU_MMAP_SIZE: The KVM and a VCPU communicate with each other using a shared region kvm_run. This api is used to get the size for this shared region kvm_run.
    - Output: size of kvm_run data structure
6. Create a struct kvm_sregs data and write the values into it from the special registers of the virtual cpu.
    - KVM_GET_SREGS: Read the values of the special registers from a VCPU.
    - Parameter:address of struct kvm_sregs variable.
    - Output: 0 on success, -1 on an error.
7. Setup page table and write the special registers include cr3 into struct kvm_sregs variable. Now write these values into the special registers of the virtual cpu.
    - KVM_SET_SREGS: Write into the special registers of a VCPU.
    - Parameter:address of struct kvm_sregs variable.
    - Output: 0 on success, -1 on an error.

8. Create a struct kvm_regs data and write the values into it from the general purpose registers of the virtual cpu.
    - KVM_GET_REGS: Read the values of the general purpose registers from a VCPU.
    - Parameter:address of struct kvm_regs variable.
    - Output: 0 on success, -1 on an error.
9. Setup instruction and stack pointer value into struct kvm_regs variable. Now write these values into the general purpose registers of the virtual cpu.
    - KVM_SET_REGS: Write into the general purpose registers of a VCPU.
    - Parameter:address of struct kvm_regs variable.
    - Output: 0 on success, -1 on an error.
10. Execute the virtual machine
    - KVM_RUN: This is used to run a virtual cpu. It takes the VCPU file descriptor as a parameter and returns an integer based on success (0) or error (-1).
    - Output: 0 on success, -1 on an error.