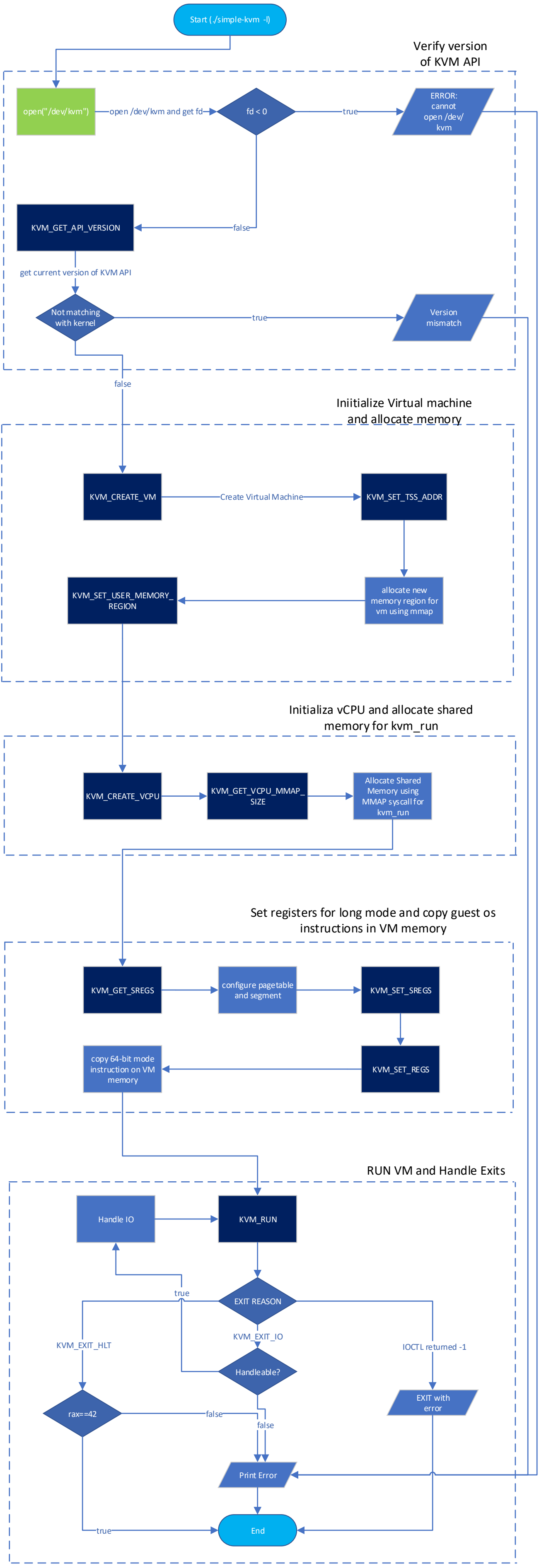
PART 1a


Start (./simple-kvm -l)
Verify version of KVM API
open("/dev/kvm")
open /dev/kvm and get fd
fd < 0
true
ERROR: cannot open /dev/kvm
KVM_GET_API_VERSION
false
get current version of KVM API
Not matching with kernel
true
Version mismatch
false
Iniitialize Virtual machine and allocate memory
KVM_CREATE_VM
Create Virtual Machine
KVM_SET_TSS_ADDR
KVM_SET_USER_MEMORY_REGION
allocate new memory region for vm using mmap
Initializa vCPU and allocate shared memory for kvm_run
KVM_CREATE_VCPU
KVM_GET_VCPU_MMAP_SIZE
Allocate Shared Memory using MMAP syscall for kvm_run
Set registers for long mode and copy guest os instructions in VM memory
KVM_GET_SREGS
configure pagetable and segment
KVM_SET_SREGS
copy 64-bit mode instruction on VM memory
KVM_SET_REGS
RUN VM and Handle Exits
Handle IO
KVM_RUN
true
EXIT REASON
KVM_EXIT_IO
KVM_EXIT_HLT
IOCTL returned -1
Handleable?
rax==42
false
false
EXIT with error
Print Error
true
End

1) KVM_GET_API_VERSION
   - Gets KVM API version number of given system.
   - IOCTL call arguments:
     - device_fd — Device File Descriptor(/dev/kvm)
     - KVM_GET_API_VERSION — Command
     - 0 — No argument needed
   - Return value:
     - integer: KVM API version number
   - This IOCT call is used to check whether underlying KVM is stable or not.

   After verifying the version we want to create our own virtual machine.

2) KVM_CREATE_VM
   - IOCTL call arguments:
     - device_fd — Device File Descriptor(/dev/kvm)
     - KVM_CREATE_VM — Command
     - KVM_VM_* — Machine Type e.g., KVM_VM_MIPS_AUTO, From simple-kvm.c to kernel module.
   - Return value:
     - integer: vm_fd — Virtual Machine File Descriptor
   - It creates an empty virtual machine, which means it doesn't have Virtual CPU and Virtual MEMORY.
   - This VM can be configured using vm_fd.

   To run the above virtual machine we have to configure vCPU and vMemory.

3) KVM_SET_TSS_ADDR
   - IOCTL call arguments:
     - vm_fd — Virtual machine file descriptor
     - KVM_SET_TSS_ADDR — Command
     - Address — Address of virtual machine memory, from simple-kvm.c to kernel module.
   - Return value:
     - integer: Zero (Success) or Non-Zero (failure)
   - x86 architecture includes a special segment called "**Task State Segment**" to store hardware contexts.
   - The CPU uses this address to find the kernel stack location in virtual machine memory.

   After configuring TSS we will allocate memory to the virtual machine using mmap syscall. This memory block will be configured using ***madvise*** to scan and identify duplicate pages by KSM(Kernel Shared Memory) daemon.

4) KVM_SET_USER_MEMORY_REGION
   - IOCTL call arguments:

    - vm_fd — Virtual machine file descriptor
    - KVM_SET_USER_MEMORY_REGION — Command
    - struct kvm_userspace_memory_region — Memory configuration data which includes memory address(hva), slot number, flags, starting physical address for this slot, and memory size.
      From simple-kvm.c to kernel module
- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call can be used to create, modify or delete guest physical memory slots.
- In this assignment, we are creating a guest physical memory slot.

We configured memory for VM, now we will configure vCPU for VM.

5) KVM_CREATE_VCPU
- IOCTL call arguments:
    - vm_fd — Virtual machine file descriptor
    - KVM_CREATE_VCPU — Command
    - 0 — No argument needed.
- Return value:
    - integer: vcpu_fd — vCPU file descriptor which is used to configure vCPU
- This IOCTL call just creates a vCPU for a given virtual machine as we are giving **vm_fd**.
- We can only add upto **max_vcpus** so, **vcpu_fd** ∈ [0, max_vcpus).

We created vCPU for a VM but now we have to assign memory for this vCPU to store and share the vCPU state with the hypervisor.

6) KVM_GET_VCPU_MMAP_SIZE
- IOCTL call arguments:
    - dev_fd — Virtual machine file descriptor
    - KVM_GET_VCPU_MMAP_SIZE — Command
    - 0 — No argument needed.
- Return value:
    - integer: vcpu_mmap_size — Size of struct kvm_run, which is a shared region
- This IOCTL call just returns the size of **kvm_run** region, this size will be used to allocate memory for that region using **mmap** syscall with **MAP_SHARED** flag.

Mmap syscall for kvm_run region will be having vcpu_fd as an argument to let KVM know that this region belongs to which vCPU.

7) KVM_GET_SREGS

- IOCTL call arguments:
    - vcpu_fd — vCPU file descriptor
    - KVM_GET_SREGS — Command
    - struct kvm_sregs — It is an output argument from kernel module to simple-kvm.c which will get current special register values into pointer variable of **kvm_sregs** structure.
- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call gets segments registers, control registers, interrupt descriptor table address(IDT) register, global descriptor table(GDT) register, Extended feature enable register.

After getting those register values in structure pointer variable we will modify the registers to configure vCPU.

8) KVM_SET_SREGS
- IOCTL call arguments:
    - vcpu_fd — vCPU file descriptor
    - KVM_SET_SREGS — Command
    - struct kvm_sregs — Configuration for special register, which will be used by the kernel module to configure vCPU. From simple-kvm.c to kernel module.
- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call in simple-kvm just configures **control registers**, **segment registers** and **efer** as we want to run a minimal OS.

After setting special registers of vCPU, we will configure general purpose registers using below APIs.
This will be our last step of configurations, after which we'll copy compiled guest-os into **vm_mem** area and now we are READY TO RUN!

9) KVM_GET_REGS
- IOCTL call arguments:
    - vcpu_fd — vCPU file descriptor
    - KVM_GET_REGS — Command
    - struct kvm_regs — It is an output argument from kernel module to simple-kvm which will get current general purpose register values into pointer variable of **kvm_regs** structure.

- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call gets general purpose registers in structure **kvm_regs.**
- vCPU has 18 general purpose registers of 64 bits each.

10) KVM_SET_REGS
- IOCTL call arguments:
    - vcpu_fd — vCPU file descriptor
    - KVM_SET_REGS — Command
    - struct kvm_regs — Configuration for general purpose register, which will be used by the kernel module to configure vCPU. From simple-kvm.c to kernel module.
- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call in simple-kvm configures **rflags, rip** and **rsp.**

Now we are ready to launch the virtual machine using **KVM_RUN** API.

11) KVM_RUN
- IOCTL call arguments:
    - vcpu_fd — vCPU file descriptor
    - KVM_RUN — Command
    - 0 — No arguments needed
- Return value:
    - integer: Zero(success) or Non-Zero(failure)
- This IOCTL call is used to actually run the VM.
- Here we are not providing any explicit argument to this IOCTL call but there is an implicit argument which is **kvm_run,** it will be used to maintain the state of vCPU.