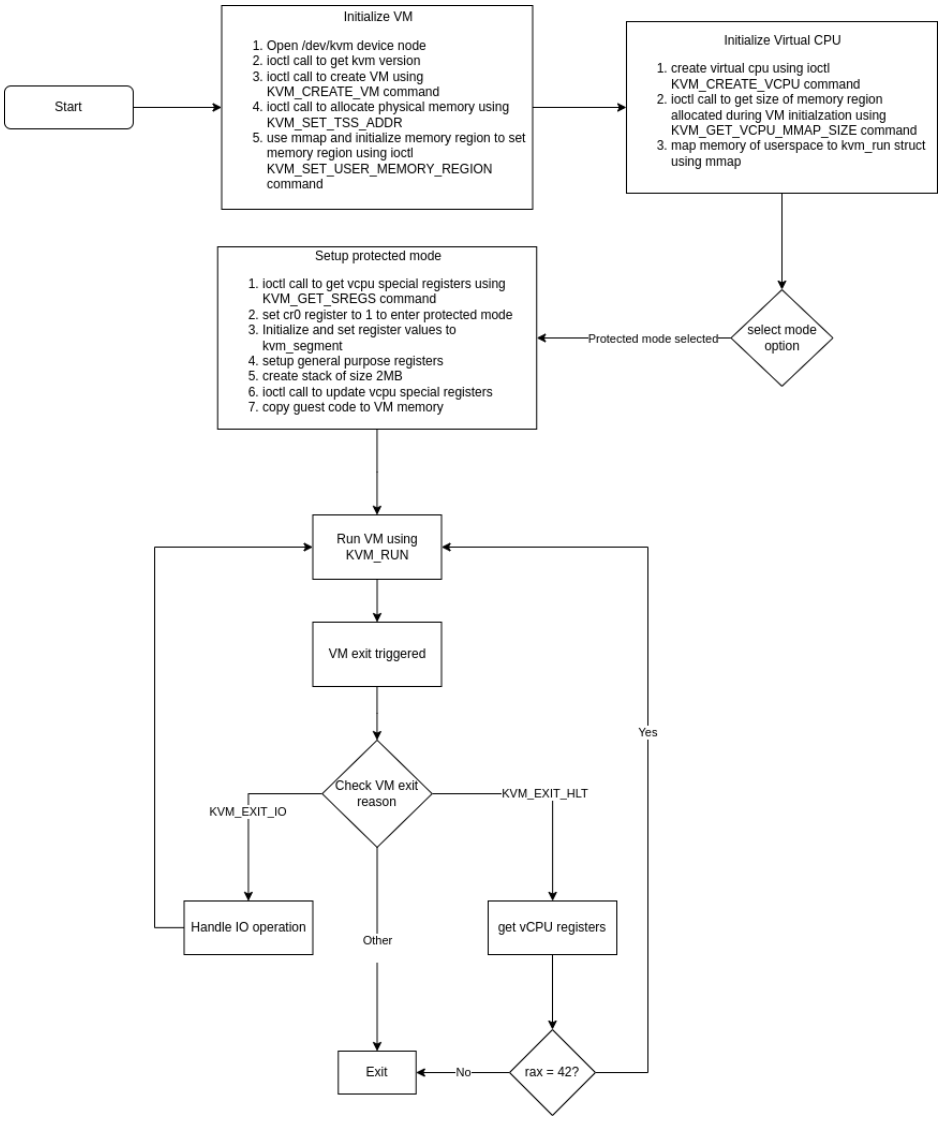
Part 1a report



1 Explanation of Code Snippets from `simple-kvm.c`

(a)

- `guest64[]` contains the program that will be executed by KVM when long mode is enabled.
- `guest64_end[]` marks the end of the program, and the program size is calculated as:

  `guest64_end - guest64`
- So guest64 and guest64_end act as start and end of a program.

(b)

- `PDE64_PRESENT` - Marks the page directory entry as present.
- `PDE64_RW` - marks entry as read/write enabled.
- `PDE64_USER` - marks page as user-mode accessible.
- `pdpt_addr,pd_addr` - marks physical address width limit.
- `PDE64_PS` - sets page size.
- `sregs->cr3 = pml4_addr;`
  set cr3 register with page map level 4 address which is a Page Directory Base Register.
- `sregs->cr4 = CR4_PAE;`
  enables Physical address extension which expands physical address space to access memory beyond 4GB memory.
- `sregs->cr0 = CR0_PE | CR0_MP | CR0_ET | CR0_NE | CR0_WP | CR0_AM | CR0_PG;`
  Enables:
  - Protected Mode
  - Monitor Co-Processor
  - Extension Type
  - Numeric Error
  - Write Protection
  - Alignment Mask

– Paging

- `sregs->efer = EFER_LME | EFER_LMA;`
  sets bits in Extended Feature Enable Register(EFER) register to enable long mode(EFER_LME) and make long mode active(EFER_LMA).

(c)

```
vm->mem = mmap(NULL, mem_size, PROT_READ | PROT_WRITE,
MAP_PRIVATE | MAP_ANONYMOUS | MAP_NORESERVE, -1, 0);
```

Allocates memory region of VM with read-write enabled.
`MAP_PRIVATE` - updates made to mapping are not visible to other process.
`MAP_ANONYMOUS` - mapping is not backed by any file.
`MAP_NORESERVE` - disable swap space reservation which does not allow mapping to be modified.

```
madvise(vm->mem, mem_size, MADV_MERGEABLE);
```

This tells kernel to enable Kernel Samepage Merging (KSM) for the pages in the range specified. it enables kernel to merge pages that have identical content into a single page.

(d)

- checks if the operation is `IO_OUT` by checking its direction and whether port used to write is `0xE9`.
- gets the pointer to kvm_run which stores the data that is to be written.
- prints the data to stdout stream which is stored at an offset - io.data_offset from base address of kvm_run struct.

(e)

```
memcpy(&memval, &vm->mem[0x400], sz);
```

Copies value at memory location `0x400` of size sz into memval used in `KVM_EXIT_HLT` during exit to check if guest os outputs correct value.