Booting a system involves several key stages:

1. **Boot Loader (GRUB)**: This manages the initial boot process.
2. **First Stage**: Typically, BIOS is invoked first.
3. **Second Stage**: This stage is responsible for setting up the environment.

**Setup Segmentation**:
- **16-bit 0...64KB**
- **Switch to 32-bit**
- **0...4GB**
- **Set up the stack**
- **Load the kernel**

**ELF into 0x1000000 (1MB)**

**Entry Point**: The kernel entry point is defined in `bootmain.c`. The boot loader `boot.asm.S` is responsible for this initial setup.

**Stack**: EIP = ESP

Physical memory layout:
- **Kernel**: 4096 (stack)
- **entry point**: The kernel is loaded starting at 0x1000000.
```
main()

mpmain()
  schedule()
  for(i;)

switch()

Esp // restore caller
// reg
ret // Eip

proc =

KSTACK
content

```

initcode.S

"init" exec

user

physreg

proc #1

end

EIP Esp
#1) ASM (x86 instruction set)
  - push
  - jump
  - call
  - pop
  - jump
  - ret

#2) ELF
  - text
  - data
  - bss
  - append

- crash
- read
- page fault
- stack

- page table
- page fault
- virtual address
- physical address
- page size
- page number
- page frame
- page table entry

- segmentation
- paging
- virtual memory
- memory management

- GCC
- LLVM
#3) Boot main
    → disk driver

#4) Address spaces
    pages + segmentation

#5) INT

#6) Context switch

2. What is GDT used for?