Operating System

Booting

1. Power on
2. CPU reset
3. POST(Power On Self Test)
4. Load boot sector into 0x7c00
5. Enter protected mode

IA-32

- IA-32 (Intel Architecture, 32-bit), as known as i386, is the title of the third generation x86 architecture.
- IA-32 was first implemented in Intel 80386 in 1985. It was the first x86 microprocessor to support 32-bit computing.

IA-32

- There are two working mode, which are real mode and protected mode.
- When we boot up the computer, CPU is working under real mode. By entering protected mode, CPU can have more powerful address capability.
- Why do we need two modes?
  Let’s review architectural history…

Intel 8086

- Intel 8086 is a 16-bit microprocessor chip
- 16-bit register
- 16-bit data bus
- 20-bit address bus
- 1MB address capability
- Physical address
  = **Segment**(16) + **Offset**(16)

Intel 8086

Example:

- 0x1000(Segment):0x1234(Offset)
  = (0x1000 << 4) | 0x1234
  = 0x11234
- 0x06EF(Segment):0x1234(Offset)
  = (0x06EF << 4) | 0x1234
  = 0x08124
- 0xFFFF(Segment):0x0010(Offset)
  = (0xFFFF << 4) | 0x0010
  = 0x00000

Intel 80386

- Intel 80386 is a 32-bit microprocessor chip
- 32-bit register
- 32-bit data bus
- 32-bit address bus
- 4GB address capability
- Still using segment:offset to represent a memory address

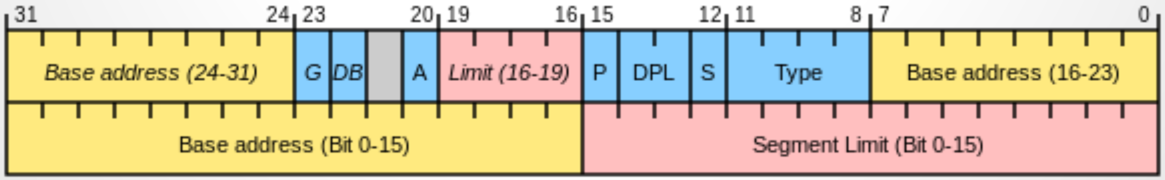
  What’s the difference?

GDT

- In real mode, an address with segment 0x1234 means a memory segment starting from 0x12340
- In protected mode, segment is an index referring to a data structure which defines the detail of the segment.
- This data structure is called **Global Descriptor Table(GDT)**.

GDT

- There is only one GDT allowed to exist in the system. In Intel architecture, the address of GDT is stored in a register GDTR.
- Shared memory and kernel memory will be described by the GDT.



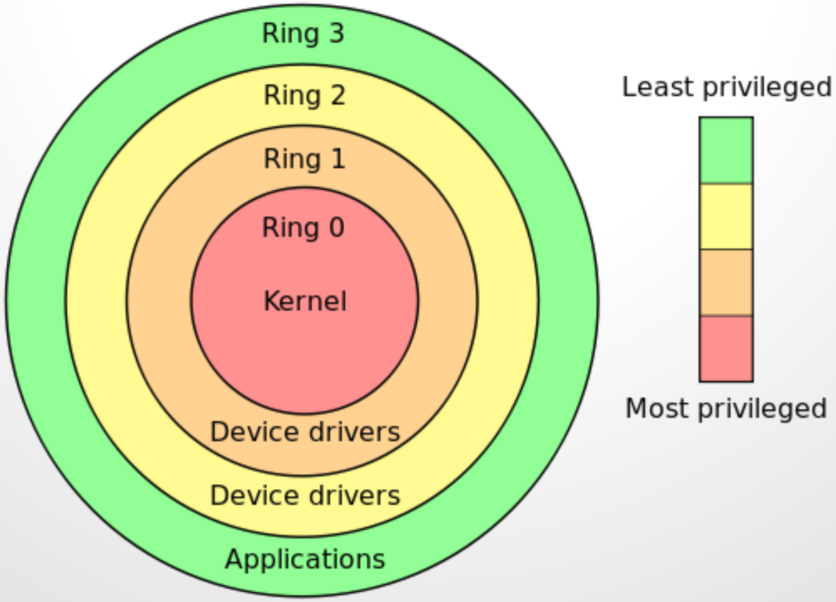
LDT

- Besides GDT, IA-32 allows programmer to create several **Local Descriptor Tables (LDT)**.
- LDT is essential to implementing separate address spaces for multiple processes. There will be generally one LDT per user process, describing privately held memory.
- LDT itself is a segment, and its segment descriptor is stored in GDT.

Privilege

Why is it called “protected” mode?

- There are four privilege levels or rings in protected mode, numbered from 0 to 3.
- Ring 0 is the most privileged and 3 is the least.



CPL, DPL, RPL

CPL(Current Privilege Level)

- The privilege level of current process. It’s stored in the lowest two bits of CS and SS.

DPL(Descriptor Privilege Level)

- The privilege level of a segment. It’s stored in the DPL bit of segment descriptor.

RPL(Requested Privilege Level)

- The privilege level of a request. It’s stored in the lowest two bits of segment selector.

CPL, DPL, RPL

Check whether a request is legal:

```
EPL = CPL > RPL ? CPL : RPL;
if (EPL <= DPL) {
    // request accepted
} else {
    // request denied
}
```

IDT

- The **Interrupt Descriptor Table(IDT)** is a data structure used by the x86 architecture to implement an interrupt vector table.
- The IDT is used by the processor to determine the correct response to interrupts and exceptions.
- Use of the IDT is triggered by three types of events: hardware interrupts, software interrupts, and processor exceptions.

IDT

Real mode:

- The IDT resides at a fixed location in memory from address 0x0000 to 0x03ff, and consists of 256 32-bit real mode pointers.
- A real mode pointer is defined as a 16-bit segment address and a 16-bit offset into that segment.

IDT

Protected mode:

- The IDT is an array of 8-byte descriptors stored consecutively in memory and indexed by an interrupt vector.
- These descriptors may be either interrupt gates, trap gates or task gates.

IDT

Protected mode:

- **Interrupt gate** will disable further processor handling of hardware interrupts, and is mainly used for handling service hardware interrupts.
- **Trap gate** will leave hardware interrupts enabled and is mainly used for handling software interrupts and exceptions.
- **Task gate** will cause the currently active task-state segment to be switched, effectively hand over use of the processor to another program, thread or process.
  (not used in Linux)

IDT

- In protected mode, the IDT may reside anywhere in physical memory.
- In Intel architecture, the processor has a special register called **IDTR** to store the base address and the length of the IDT.
- When an interrupt occurs, the processor multiplies the interrupt vector by 8 and adds the result to the IDT base address.
- The 8-byte descriptor at the result address location is loaded and actions are taken.

Summary

- Segment addressing
- Descriptor table
- Privilege levels
- Interrupt vector